SUDEEP GHIMIRE

Mestre em Lógica Computacional

Self-Evolutionary Cyber Physical Systems: Leap towards smart CPS

Dissertação para obtenção do Grau de Doutor em

Engenharia Electrotécnica e de Computadores

Orientador:	Prof. Doutor Ricardo Luís Rosa Jardim Gonçalves, Professor Associado com Agregação da Faculdade de Ciências Tecnologia da Universidade Nova de Lisboa
Coorientador:	Prof. Doutor António Carlos Bárbara Grilo, Professor Auxiliarda Faculdade de Ciências Tecnologia da Universidade Nova de Lisboa

Júri:

Presidente:	Prof. Doutor Fernando Jose Pires Santana
Arguentes:	Prof. Doutor Jose Mendes Machado
	Prof. Doutor Yacine Ouzrout
Vogais:	Prof. Doutor Paulo da Costa da Luis da Fonseca Pinto
	Prof. Doutor Ricardo Luis Rosa Jardim Goncalves
	Prof. Doutor Joao Carlos da Palma Goes
	Prof. Doutor Celson Pantoja Lima
	Doutor Joao Filipe dos Santos Sarraipa

FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

December, 2016

Copyright

Dedicated
To my family
-The fabulous four

With Love and Devotion I dedicate to thee.

ACKNOWLEDGEMENTS

It has been a long and challenging journey, yet very fruitful and satisfying. The completion of this undertaking could not have been possible without the support, participation and assistance of numerous individuals, whose name my not all be enumerated. Their contributions are sincerely acknowledged.

My deepest acknowledgements to my parents, for taking up the hardest parts so that things were much easier on my sides and I know how proud they are of my achievements.

To, my wife for understanding and being on my side even when we had to pass through difficult times.

My heartfelt appreciations to my supervisor- Prof. Ricardo Luís Rosa Jardim Gonçalves and co-supervisor Prof. António Grilo for their support and guidance. I am very indebted for the time, effort and patience that they have invested leading me to achieve my goals. The constant feedbacks and encouragements during this research work has been the most important catalyst for the entire period.

To, my colleagues at GRIS, my research group, with whom not only I have shared work space but also many interesting ideas and moments of fun and laughter. Special shout to fellow researchers Fernando and Tahereh for fruitful discussions that resulted to publications and to new ideas.

I expressed grateful gratitude towards both of my institutions, UNINOVA and FCT for hosting out research and wonderful academic environment that allows us to pursue bigger goals.

To, all relative, friends and strangers who have shared their support either morally or emotionally at hard time, thank you all!!

ABSTRACT

Change over time is a fact. The world is characterized by ongoing processes of development, formation, and growth in both natural and human-created systems. For instance the present diversity of plant and animal life arose from the earliest and most primitive organisms through gradual changes. This process of gradual changes over generations is termed ***evolution*** and has been long studied in biology which basically leads to the fact that complex, natural systems are not created all at once but must instead evolve over time. Theory of evolution and the fact that evolutionary processes are ubiquitous and critical for social, educational, and technological innovations is accepted by researchers working in different domains. In the scope of this dissertation, the major driving force behind the study of evolutionary principles is solving real-world problems that have frequent changes in the scope of problems or the surroundings of the problem.

Cyber Physical Systems (CPSs) are systems that comprises of integration of computation and physical processes. In CPS, computation and communication involves interaction with physical environment to add new capabilities and characteristics to systems. However, the integration of physical component increases the factor for unreliability to the overall system because of the unpredictable behaviours of the physical world. This makes CPS as dynamic systems that are adaptive to the changes that will occur at runtime based on the uncertainties in the physical world. In such types of systems "*evolution*" is an intrinsic feature. Studying such types of systems and developing suitable solutions has been the goal of this research work. The results thus obtained are presented in this dissertation. Furthermore, the research work presented here provides research results in the direction of self-evolutionary CPS which can be the base for mission critical systems like air-traffic control, disaster detection and also for business processes which are highly dependent on real-time data. In such scenarios, self-evolving systems will be prepared to automatically detect changes in the external circumstances that can affect internal conditions, take mitigation measures (*adaptation loop*) and learn from the changed scenario based on the acquired understanding of system behaviour (*learning loop*) so that whole system can take long term mitigation actions to avoid future cases of failure (*evolutionary loop*).

The mission hereby assumed is to provide theoretical and technological solution that can be used for defining sensorial and behavioural modelling for CPSs which can be coupled with the model of evolutionary machines in order to achieve evolutionary CPS. The self-characteristics are obtained via framework for dynamic re-configuration and methodology for injecting acquired knowledge into the evolutionary model. The presented research results are validated by some industrial use-cases and at the same time provide some important base for further research.

KEYWORDS

Evolution; Cyber Physical Systems; Sensorial modelling; Behavioural modelling; Dynamic re-configuration; Adaptation; Learning.

RESUMO

As mudanças ao longo do tempo são um facto. O mundo é caracterizado por processos de desenvolvimento em curso, formação e crescimento de ambos os sistemas, tanto natural como centrado nos humanos. Por exemplo, a actual diversidade de plantas e vida animal surgiu dos organismos primitivos iniciais através de mudanças graduais. Este processo de mudanças graduais ao longo das gerações é chamado de evolução e tem sido demoradamente estudado em biologia o que basicamente conduz-nos ao facto de complexos, sistemas naturais não são criados logo de uma vez mas ao invés evoluem ao longo do tempo. No âmbito desta dissertação, a principal motivação por trás do estudo dos princípios evolucionários é a resolução de problemas do mundo real os quais sofrem mudanças frequentes no âmbito dos problemas ou dos problemas das suas fronteiras.

Sistemas Cyber Físicos (CPSs) são sistemas que compreendem a integração de computação com processos físicos. Em CPS, computação e comunicação envolvem interacção com o ambiente físico para adicionar novas competências e características aos sistemas. No entanto a integração de componentes físicos diminui o factor de fiabilidade do sistema como um todo devido ao comportamento imprevisível do mundo físico. Isto torna os CPS em sistemas dinâmicos que que são adaptativos às mudança e que vão ocorrer a quando da execução baseadas nas incertezas do mundo físico. Em tais tipos de sistemas, "evolução" é uma característica intrínseca. Além disso, o trabalho de investigação aqui documentado apresenta resultados de investigação na direcção dos auto-evolucionários CPS os quais podem ser a base para sistemas de missões criticas tais como o controlo de tráfego aéreo, a detecção de desastres e também para processos de negócios que são altamente dependentes de dados em tempo real. Em tais cenários, os sistemas auto-evolutivos estarão preparados para detectar automaticamente alterações nas circunstâncias externas que podem afectar as condições internas, tomar medidas para mitigar (ciclo de adaptação) e aprender pelo cenário modificado baseado no conhecimento adquirido sobre o comportamento do sistema (ciclo de aprendizagem) de forma a que o sistema como um todo possa tomar acções de mitigação de longo termo de forma a evitar futuros casos de falha (ciclo evolucionário).

A missão aqui assumida é a de providenciar soluções teóricas e tecnológicas que possam ser utilizadas para definir a modelação sensorial e comportamental para CPSs os quais podem ser acoplados com o modelo de maquinas evolucionárias de forma a alcançarmos os CPS evolucionários. As auto-características são obtidas num quadro de dinâmica reconfiguração e metodologias para injectar conhecimento adquirido num modelo evolucionário. Os resultados da investigação apresentada são suportados por alguns casos de uso industriais e em simultâneo constituem uma base importante para um domínio de investigação interessante para a investigação futura.

PALAVRAS CHAVE

Evolução; Sistemas Cyber Físicos; Modelação Sensorial; Modelação Comportamental; Dinâmica Reconfiguração; Adaptação; Aprendizagem.

GLOSSARY

5C	Connection, Conversion, Cyber, Cognition, and Configuration
AAN	Artificial Neural Network
AI	Artificial Intelligence
BEM	Basic Evolutionary Machine
BRCEM	Basic re-configurable Evolutionary Machine
C2NET	Cloud Collaborative Manufacturing Networks
CPS	Cyber Physical Systems
DEECo	Dependable Ensembles of Emerging Components
EA	Evolutionary Algorithms
EC	Evolutionary Computation
EFA	Evolutionary Finite Automaton
EHFA	Evolutionary Hybrid Finite Automaton
EP	Evolutionary Programming
ES	Evolutionary Strategies
ETM	Evolutionary Turing Machine
FInES	Future Internet Enterprise System
FITMAN	Future Internet Technologies for MANufacturing industries EC Contract No. 604674
FI-WARE	Future Internet- WARE Core Platform of the Future Internet, EC Contract No.: 285248
FOF	Factories of the Future
FPGA	Field-Programmable Gate Array
FSM	Finite State Machine
GA	Generic Algorithms
HA	Hybrid Automata
HTTP	Hypertext Transfer Protocol
IOP	Interaction-Oriented Programming
IoT	Internet of Things
KE	Knowledge Engineering
KRR	Knowledge Representation and Reasoning

MBD	Model Based Design
MBE	Model Based Engineering
MSM	Master-slave Model
MoC	Models of Computation
NGN	Next Generation Network
NIST	National Institute of Science and Technology
OS	Operating System
PIM	Platform Independent Model
PLE	Product Line Engineering
QM	Quality Management
RE	Requirement Engineering
REST	REpresentational State Transfer
RETOS	Resilient, Expandable, and Threaded Operating System
RFID	Radio frequency Identification
ROAD	Role-Oriented Adaptive Design
RS	Reinforcement Sensing
SO	Self-Organizing
SOA	Service Oriented Architecture
SoS	System of Systems
V&V	Verification and Validation
WPAN	Wireless Personal Area Network
WSN	Wireless Sensor Network
WWW	World Wide Web

INDEX

LIST OF FIGURES

LIST OF TABLES

SECTION I: INTRODUCTION AND RESEARCH PLAN

This is the first part of the dissertation and builds up the foundation for the research work along with the methodology followed for the research development. Within this section the motivation for developing the present work is explained along with the identification of major research domains, gaps and aimed contributions. It also describes the methodology that guided the present research work that provides contributions to knowledge in the presented research areas. Envisaging that goal, Section I is divided in two chapters. This section is divided into two chapters: ***Chapter 1: Introduction*** *and* ***Chapter 2: Research Methodology***

Chapter 1: *Introduction provides necessary sections for overall understanding of the research work with necessary motivation and the vision that has lead towards this research work. Thorough observations in the current landscape of cyber physical systems research is presented in the beginning of the chapter. While the second chapter is dedicated for identification of some important research areas related to the overall research work along with gaps and aimed contributions. This chapter ends with description of the dissertation of organisation and by the end of the chapter it is expected that the reader will have the overall understanding of the research problem.*

Chapter 2: *Research Methodology, presents the work plan that guided the research work developed, describing the different steps of the scientific methodology. This chapter presents the research questions that have been raised during this research work along with a quick discussion on some rational thoughts that have been raised during the beginning of the research work. The chapter ends with the hypothesis that has guided the work leading towards this thesis.*

On the whole by the end of this section, it is expected that for the reader has the idea on why this work started, what was the pursued objective, what was the research methodology adopted along with the way the studies and experiences are presented in this document.

1. INTRODUCTION

Some people would claim that things like love, joy and beauty belong to a different category from science and can't be described in scientific terms, but I think they can now be explained by the theory of evolution.

-Stephen Hawking

In this chapter the main objective is to provide the justification for the need for studying evolution in the context of cyber physical systems. Introduction aims to build the understanding of the subject, motivation and pave a long term vision that will lead towards the research work discussed in the following sections. Section 1.1 provides the discussion on the current state of play in the scope of cyber physical systems including some important characteristics of such systems which gives the space for analysis of the present situation and mark some high level challenges. This builds up to the explanation of motivation and overall vision as discussed in sections 1.2 and 1.3 respectively. Section 1.4 is an overview of important research areas with the identification of some scientific and technical challenges. The identified research areas form the base for the literature review presented in details in Section II. This chapter ends up with the outline of the dissertation to provide a guideline for following this dissertation.

The world is characterized by evolution—that is, it is composed of ongoing processes of development, formation, and growth in both natural and human-created systems. It has been long studied in biology that complex, natural systems are not created all at once but must instead evolve over time. Theory of evolution and the fact that evolutionary processes are ubiquitous and critical for social, educational, and technological innovations is accepted by researchers and common people equally. The major driving forces behind the evolution of systems are their use by communities of practice in solving real-world problems with frequent changes in the scope of problems as well as the changing nature of the world. To have a technological perspective of necessity for research on evolutionary computation we can take the fact -the basic assumption that complete and correct requirements can be obtained at some point of time is theoretically and empirically wrong [1]. System requirements collaboratively evolve through an iterative process of consultation between end users [2] and software developers and specification errors often occur when the technical and business team do not have sufficient application domain knowledge to interpret the customer's intentions from the requirement statements [3].

In the recent time there has been a rapid research and technological advances in embedded systems supported by developments in wireless communications and increasing availability of sensors, actuators, and mobile devices. This has led to new ubiquitous computing paradigm that facilitates computing and communication services all the time, everywhere providing added values over real-time data. This emerging paradigm is changing the way we live and work today and also giving rise to more complex systems. Global and localized networks, users, sensors, devices, systems and applications can seamlessly interact with each other and even the physical world in unprecedented ways. This clearly depicts the development of “systems-of-systems” that

interact with real-world environment or of "systems" that have equally close connection with both the physical and the computational components. The new advancement in technology opens up the need for systems to provide seamless integration of physical world with the digital world. Such systems are termed "*Cyber Physical Systems*" (CPS), which act independently, co-operatively or as "*systems-of-systems*" composed of interconnected autonomous systems originally independently developed to fulfil dedicated tasks. In general, CPSs refer to the next generation of engineered systems that require tight integration of computing, communication, and control technologies to achieve stability, performance, reliability, robustness, and efficiency in dealing with physical systems of many application domains [4].

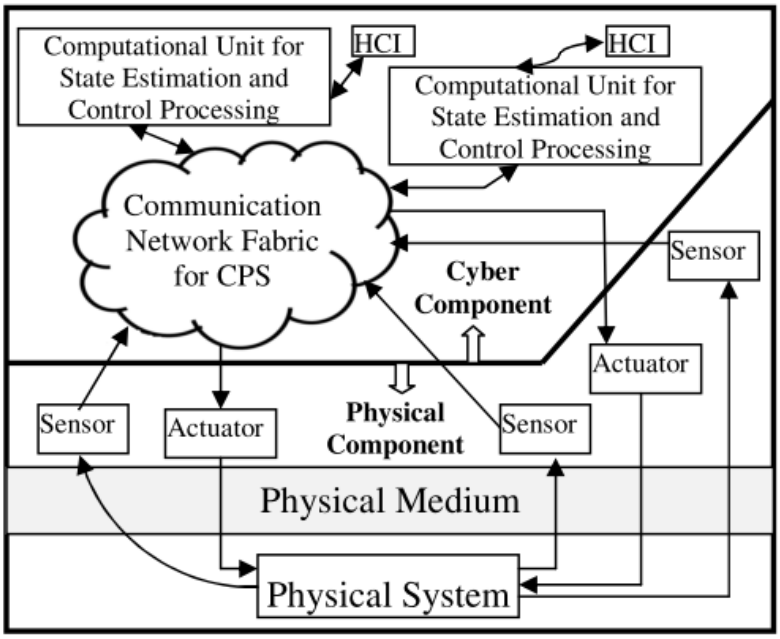


Figure 1-1 Nuts and Bolts View of a Cyber Physical System

The physical component of CPS is composed of physical devices with capabilities such as integrated networking, information processing, sensing and actuation, can operate in close relation with the physical environments as depicted in Figure 1-1 taken from [4]. These abilities can lead towards the realization of systems that can be responsive to real-time changes in the environment. While, the cyber component comprises of computational unites that has proven capability for high end computation, storage and analytics. Such tightly coupled cyber and physical systems that exhibit this level of integrated cooperation and intelligence are characteristics of CPS. The computational and physical processes of such systems are tightly interconnected and coordinated to work together effectively, often with humans in the loop [5]. Tight coupling with the unpredictably changing physical component injects necessity for evolutionary behaviour in CPS. CPS workshop report by NIST summaries this scenario as "*System uncertainty must be characterized and quantified to understand the implications of the inputs and their variability on system operation that leading towards the need for an evolutionary system that can adapt to changing naturalistic inputs.*" [6].

1.1. OBSERVATION

The growing trend toward computational intelligence, automation, and control for complicated but well-defined tasks or processes, especially when demands or constraints are not amenable to human intervention paves the path for application of CPS. For example, automatic collision systems could detect moving objects and respond faster than a human operator. Unmanned CPS could be used to reduce the risk to human life by detecting mines, exploring volcanoes, or conducting otherwise hazardous tasks. Machines driven by a computer do not suffer fatigue and may be more precise than is humanly possible. In future CPS could make possible concepts only imagined today, such as unmanned tours to the moon, bionic suits, and automated largescale indoor agriculture systems [5]. At the same time by considering the dynamic nature of the run-time environment of the CPSs, evolution over time can be observed leading towards the realization of CPSs that are not only capable to co-exist in ever changing environmental conditions but also have characteristics for self-adaption , self-configuring and eventually self-evolutionary. These classes of CPSs can be termed as self-evolutionary CPS (eCPS).

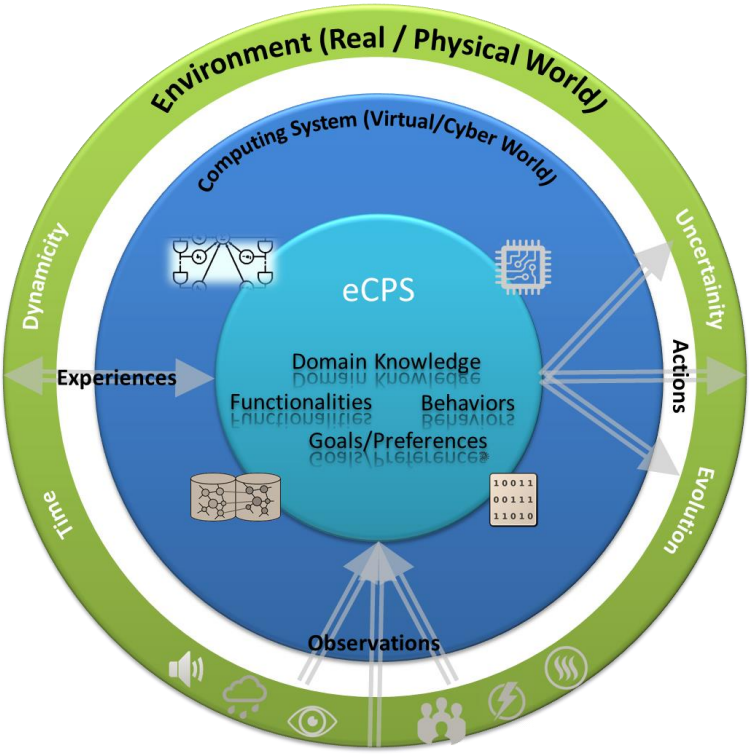


Figure 1-2 Overall paradigm of CPSs

The overall paradigm of eCPS is as depicted in Figure 1-2, which captures the most important characteristics of the real-world environment i.e. dynamicity, uncertainty, time and evolution. The CPS systems continuously interact with the environment from which observations are collected that lead towards the generation of actions and exchange of experience. Cyber physical systems to be developed will have functional/non-functional and behavioural specifications which are enriched with domain knowledge capturing the

system domain. At the same time system goal and preferences are specified to achieve the realization of some processes such as manufacturing processes, business processes etc. Technically CPSs are composed of sensor-actuator networks, embedded/real-time systems, desktop/laptop etc., none of them could exclusively be considered as a CPS. CPS is a dynamically reorganizing and reconfiguring control system with high degree of automation, complexity at multiple spatial and temporal scales, and control loops closed at all scales [7]. They are also systems with entities networked at multiple scales possibly with cyber capability at each physical component and dependable and certifiable (secure) operations. In short, high degrees of complexity and tight coupling and coordination between system's computational and physical entities through networked communication characterize cyber physical systems [4]. European Roadmap on Research and Innovation in Engineering and Management of Cyber Physical Systems [8] has identified that CPSs exhibit the features as marked in

Table 1-1 Most important features of CPS

I.	*Large, often spatially distributed physical systems with complex dynamics*
II.	*Distributed control, supervision and management*
III.	*Partial autonomy of the subsystems*
IV.	***Dynamic reconfiguration of the overall system on different time-scales***
V.	***Continuous evolution of the overall system during its operation***
VI.	***Possibility of emerging behaviours***

Above mentioned features pave the path for research work for robust, resilient and smart CPSs. In the scope of this PhD work, the last three features (highlighted in bold) are of great focus.

Dynamic reconfiguration, i.e. the frequent addition, modification or removal of components is an intrinsic phenomenon in CPS, thus requiring extensive studies for the realization of eCPS. This aspect is important for the handling of faults and the change of system structures and management strategies following changes of demands, supplies or regulations; fault detection and handling of errors or abnormal behaviours etc. Since the working environment of CPS exhibits unpredictable changes, the performance of overall system is strongly affected by the impact of unforeseen events and outer influences that require non-continuous actions.

Industrial CPSs are large systems that operate and are continuously improved over long periods of time such that the hardware (real physical hardware) infrastructure have longer operation life and new functionalities or improved performance have to be realized with only limited changes of parts of the overall system. While, the changes in the business and or technological contexts can bring abrupt need for components to be modified, added and or the scope of the system may be extended or its specifications changed. So engineering to a large extent has to be performed at runtime to address the changes in

requirements change during operation. Once rolled out into production, operating and maintaining a complex CPS requires a good knowledge of the "as-deployed-and-configured" system's physical, functional and behavioural configuration. Thus, CPSs exhibit the run-time behavioural changes and it is an important aspect to address via. the research across ***continuous evolution of systems***.

In complex systems with autonomous subsystems, it is expected for the overall system to exhibit significant diversity in their behaviours governed by local working environment of each subsystem. Interactions between subsystems with local diversity can lead towards emergent behaviours. ***Emerging behaviour*** should be distinguished from cascades of failures. However, if series of faults lead to instabilities and system goes through functional failures and/or correction cycles then this can be called emerging behaviour. Emerging behaviours are thus unpredicted behaviours during system development but faced during the run-time often across a long period of operation. Emerging behaviour should be addressed both from the side of system analysis – under which conditions does emerging behaviour occur – and from the side of systems design – how can sufficient resiliency be built into the system such that local variations, faults, and problems can be absorbed by the system or be confined to the subsystem affected and its neighbours and do not trigger cascades or waves of problems in the overall system

1.1.1. *THE PRESENT*

Cyber physical systems are starting to be seen as technology advancement of crucial importance as they represent some of the most important infrastructures, e.g. systems for the generation and distribution of electric energy, drinking water and gas, rail, road, air and marine transportation systems and their elements, and large industrial production processes. Cyber Physical Systems (CPSs) are bridging the virtual and physical world enabling us to communicate with physical objects with application in various domains, such as transportation, health-care, manufacturing, agriculture, energy, defense, aerospace, buildings and public environments. Foundations for Innovation in Cyber Physical Systems [5], has clear analysis on applications of CPSs in different domains like smart manufacturing, transportation and mobility, energy, healthcare etc. Some highlighted benefits of CPS are Intelligent controls, Process and assembly automation, Robotics working safely with humans etc. in the industries of next generation which will have impact with enhanced global competitiveness; high tech manufacturing and greater efficiency, agility, and reliability. Thus, the domain of industrial systems is increasingly changing as it adopts emerging Internet based concepts, technologies, tools and methodologies. The rapid advances in computational power, coupled with the benefits of the Cloud and its services, has the potential to give rise to a new generation of service-based industrial systems whose functionalities reside in-Cloud (Cyber) and on-devices and systems (Physical).

As we move towards an infrastructure that is increasingly dependent on monitoring of the real world, timely evaluation of data acquired and timely applicability of management (control), several new challenges arise. Future factories [9] are expected to be complex System of Systems (SoS) that will empower a new generation of applications and services that, as yet, are impossible to realise owing to technical and financial limitations. The European IMC-AESOP project is a visionary undertaking by key industrial players such as Schneider Electric, SAP, Honeywell and Microsoft, investigating the applicability of cloud-based Cyber Physical Systems (CPS) [10] and Service Oriented Architectures (SOA) [9] in industrial systems, such as monitoring of paper production machines by FluidHouse (Finland) and lubrication monitoring and control systems in mining industry by LKAB (Sweden).

The Factory of the Future (FoF) [9] will rely on a large ecosystem of systems in which large scale collaboration will take place. Additionally, it is expected that CPS will harness the benefits of emerging Cloud Computing, including resource-flexibility and scalability. This not only has the potential to enhance the functionality of CPS, but will also enable a much wider consumption of CPS's data and services. The result will be a highly dynamic flat information-driven infrastructure that will empower the rapid development of better and more efficient next generation industrial applications whilst simultaneously satisfying the agility required by modern enterprises.

It looks very clear that the next industrial revolution will be guided by the technological advancement in CPSs and related domains. The future applications of CPS are more transformative than the IT revolution of the past three decades [11]. The next generation of CPSs should be prepared for the exponential growth of system and unexceptional dynamics of run-time environment under unforeseen conditions, thus it is necessary to integrate working and learning behaviours intrinsically in the system, it will be possible to create computational knowledge and/or memories that include mechanisms to capture and represent task specifications, work artefacts, group communications, adaptation mechanisms etc. These knowledge/memories facilitate system learning by supporting the evolution, reorganization, and sustainability.

1.1.2. *CHALLENGES*

CPSs are physical and engineered systems, which can monitor and control the physical environment. Typically, they possess a computing and communication subsystem being interconnected with each other and further integrated with other CPSs, ISs and the Internet in general. In defining the research challenges of CPS, major challenges arise from the heterogeneity of the system components forming a CPS, interconnectivity, system dependencies and the lack of a comprehensive data analysis along with security concerns at cyber, physical and communication layers. At the same time as CPS development requires different disciplines and includes knowledge specific to different research domains, an essential part of this research efforts is to locate and use synergies

between various research results and industrial requirements and use-cases. It is important to state that in the scope of this research work are focusing on data-collection and analysis, information offering and integration of consumer electronics such as mobile devices and wearables. We claim that interdisciplinary research a requirement to establish save, transparent and modern CPSs.

However, even though not an important research challenge addressed by this research work, security is an important aspect for consideration for realization of robust and smart CPSs. As the interaction between the physical and cyber components increases, the physical systems become increasingly more susceptible to the security vulnerabilities in the cyber system. Sensors, which form the lowest layer of CPSs, have unique characteristics that warrant novel security considerations: the geographic distribution of the devices allows an attacker to physically capture nodes and learn secret key material, or to intercept or inject messages; the hierarchical nature of sensor networks and their route maintenance protocols permit the attacker to determine where the root node is placed. Perhaps most importantly, most sensor networks rely on redundancy (followed by aggregation) to accurately capture environmental information even with poorly calibrated and unreliable devices [12]. Besides the security vulnerability in terms of hacks and attacks, another important aspect for consideration is trust and validity. Different types of vulnerabilities, attacking models and adversary types in the scope of CPS are well explained in the research paper [13], along with discussion on a set of challenges and research problems that need to be resolved in the future.

The future applications of CPS are more transformative than the IT revolution of the past three decades. Real-time networked information and pervasive sensing, actuating, and computation are creating powerful opportunities for systems integration [11]. Future CPS have many sophisticated, interconnected parts that must instantaneously exchange, parse, and act on detailed data in a highly coordinated manner [5]. Considering the critical nature of CPS, it's an important challenge to cope up with multiscale, multi-layer, multi-domain, and multi-system integrated infrastructures will require new foundations in system science and engineering. Another important root level necessity and challenge for robust CPS is definition of standards and protocols so as to ensure that all interfaces between components comprising CPS are both composable and interoperable.

Now, considering the scope of the thesis, the first important challenge is to have an evolutionary system is to empower the individual devices with service oriented infrastructure. This will enable each single unit of the complex system to:

i. Expose their functionalities as services, and

ii. Empower them to discover and invoke services of other components to complement their own functionalities.

This requires the need for modelling environment where each component is independently modelled and run-time environment that supports dynamic reconfiguration. To achieve evolutionary systems, some important research challenges to be addressed are:

Dynamic reconfiguration: Since CPS doesn't work in strictly controlled environment; there is a massive need for detecting different types of situations quickly and respond/react to such situations. This is necessary not only to have adaptation over system behaviour but also to formulate fail prevention or utilize soft-fail mechanisms which can incorporate resiliency and fault tolerance at the systems level. For this purpose, living cells with their multiple metabolic pathways are an example of a system that has optimized its ability to reconfigure itself to cope with changing conditions (availability of nutrients and other external factors) by keeping many options (metabolic pathways) intact and being able to switch between them. Evolutionary computing also provides background for selecting optimal configuration from different possible configurations. But, the challenge that remains open or has not been explored yet is how, reconfiguration across both physical and cyber domain can be achieved in CPS, without disturbing the consistency of both the world.

Continuous evolution: System development and deployment is a well-defined process and the activities and actors involved at each phase cooperate and coordinate in a controlled environment. Control factors are often standards, protocols, methodologies etc. But the after the deployment phase, the operational or run-time phase is rather dynamic and uncertain. And, these factors are even higher in CPS. Take the CPS in dynamic business domains such as collaborative manufacturing, construction etc. the factors for changing run-time environment increases even higher. In these types of scenarios, the experience gained in run-time environment must also take care of the implementation of engineered changes in a running system. Thus, systems can undergo continuous evolution rather than develop and use paradigm of traditional software. This issue has been studied from software engineering point of view, but has been less explored in the CPS domain, thus opening challenges to define methodology that can integrate continuous evolution in all the phases of system development. Also, it requires formal engineering models that can be used for automated investigation of options for modifications as well as improved operational policies without modifications. The engineering of system of systems requires methods and tools that can be used seamlessly during design as well as operation (design-operations continuum).

Emergent Behaviours: From the two challenges discussed before, it's clear that dynamicity of CPS shows emergent behaviours, which need to be handled. Formal verification (e.g. assume/guarantee reasoning) as well as dynamic stability analysis for large-scale systems are possible approaches to prove the non-existence of unwanted emerging behaviours as well as detection of new possible emergent behaviours. The new behaviour can be analysed with evolutionary algorithms to check if they still meet the goal or not. Also, in CPS the behaviour of the large coupled physical part of the system must be modelled, simulated and analysed using methods from continuous systems theory. In technical systems, emerging behaviours usually are seen as problematic as a predictable behaviour of the system is preferred. In large systems with subsystems that show significant diversity in their behaviours, the formation of stable structures on a

higher level due to the interactions between the subsystems despite their local diversity is very important and an interesting research challenge.

1.2. MOTIVATION

Engineering of safety-critical CPS requires integration of heterogeneous modelling methods from different disciplines. It is often necessary to view this integration from the perspective of analyses – algorithms that read and change models. Although analytic integration supports formal contract-based verification of model evolution, it suffers from the limitation of analytic dependency loops. Dependency loops between analyses cannot be resolved based on the existing contract-based verification. This paper makes a step towards using rich architectural description to resolve circular analytic dependencies. We characterize the dependency loop problem and discuss three algorithmic approaches to resolving such loops: analysis iteration, constraint solving, and genetic search. These approaches take advantage of information in multi-view architectures to resolve analytic dependency loops.

Besides these recent developments, the base of CPS i.e. embedded systems has been used in the automotive industry as early as 1970. Since then, new requirements, functionalities and networking have dramatically increased the scope, capabilities and complexities of CPS. This has created needs to bridge the gaps between the separate CPS sub-disciplines (computer science, automatic control, mechanical engineering, etc.) and to establish CPS as an intellectual discipline in its own right. The development of a CPS involves many stakeholders who are interested in different aspects of the system. Consider for example the development of an embedded control system such as an advanced driver assistance system (ADAS) (e.g. adaptive cruise control). Building such a system naturally involves multiple engineering disciplines, dealing with requirements, control design, software development, hardware development, etc.

Data scientists have defined a five-level architecture also called 5C architecture for CPS. The layers are divided based on the tasks involved in CPSs mainly in the manufacturing domain. The pyramid-shaped visualization is useful to represent the way data passing to higher levels gets reduced in size while the value of the information raises. Figure 1-3 taken from [14], provide a clear vision for CPS. Compared with DIKW Pyramid (Data, Information, Knowledge, Wisdom), which is presented in [15,16] and is an well accepted pyramid by many data scientists, the five-level architecture specifically focuses on how to enable physical machines to utilize Data and Information to create Knowledge and Wisdom leading towards realization of self-* behaviours in CPSs such as self-adaptive, self-configuring, self-organization, self-management, self-evolutionary etc. In the scope of this thesis work, in comparison with the 5C architecture, we envision to provide specific scientific contribution in the top two layers i.e. ***Cognition*** and ***Configuration***. But the reference framework for eCPS and technical implementation for the validation of the research work will also consider ***Smart Connection*** and ***Data to Information*** layers.

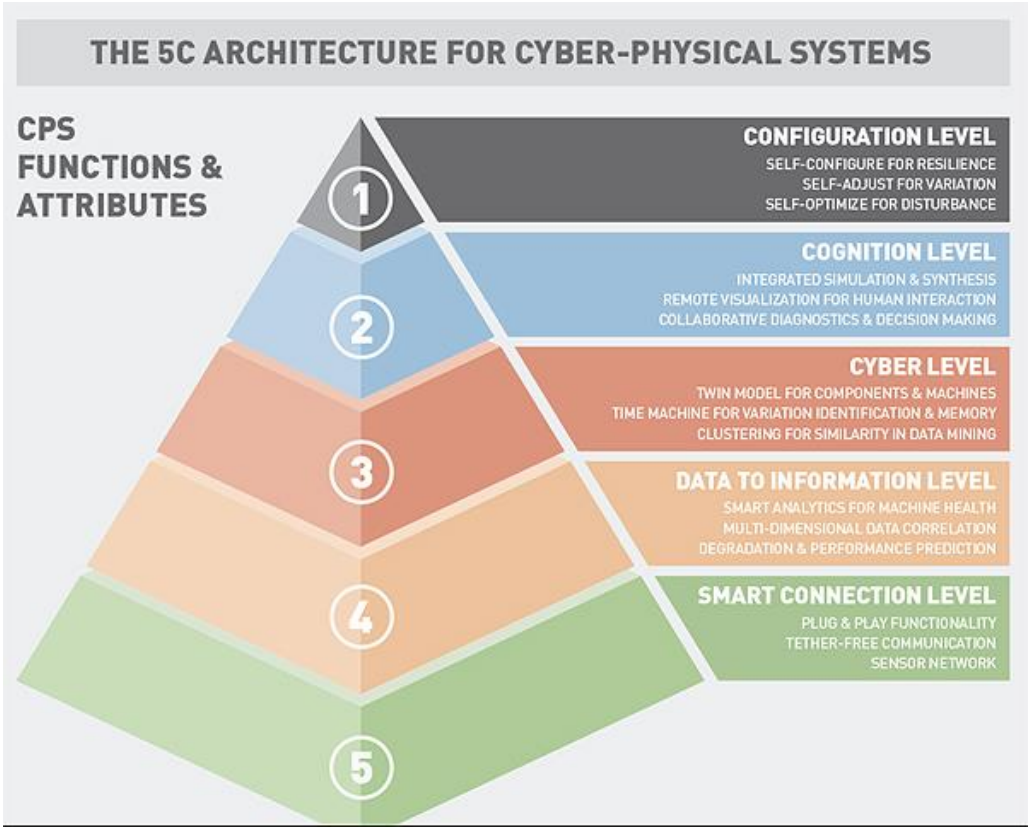


Figure 1-3 5C architecture for implementation of Cyber Physical System

Conceptual approach to build evolutionary systems involves designing the elements of a system to find by them the solution of the problem. Like this, when the problem changes, the elements are able to dynamically find a new solution. We can say that such a system self-organizes. Even when the concept of self-organization [17] is very promising to solve complex problems, and has been used for more than half a century, it remains somewhat vague, and it is not widespread. The aim of this work is to enhance our understanding of self-organization, and to exploit it to build systems that will be able to cope with complex problem domains. With the experience gained by building such systems, our understanding of them is also increased. There is as yet no general methodology to design and control self-organizing systems. This thesis is a step towards developing one, providing new insights to build systems able to solve complex problems.

The role of a control mechanism in cybernetics [18] is to regulate the variables of a system against perturbations within a viability zone. Thus, the control forces the system into a certain region of the state space. For example, a thermostat controls the variable 'temperature' to be in a viability zone defined by the user. However, it becomes very complicated to steer the variables of a complex system due to inherent nonlinearities[19].

Based on these observations, it raises some interesting questions:

- *What if a machine can learn from its own history and also other machines? E.g. What if a wind turbine can learn from its peers within the same wind farm so that its condition and maintenance requirement can be quickly generated?*

- *What if a machine can learn from human knowledge and operations to improve its intelligence for error prevention?*
- *What if the machine learns from the above steps and evolve over time so that they become more robust (and/or immune) to harsh working conditions?*
- *How can the nonlinearities of the system be monitored to define learning and evolutionary algorithms?*
- *How can we develop systems that can sense and react to the environment and automatically enforce or retract some behaviour?*

These are the questions that have motivated for the research plan that is presented. It is quite obvious that there is no universal answer, but the aim is to increase the understanding of systems of high complexity and to define methodologies for realization of such systems.

1.3. VISON

This research work has been inspired by the theory of evolution and it's a pursuit for understanding the principles of natural evolution and study the correlation for contribution towards evolutionary computation. In pursing nature inspired evolutionary computation, it is also considered the vision of a new generation of cyber physical systems i.e. self-evolutionary cyber physical system (eCPS).

One of the main visions is formulating the methodology for realization of self-evolutionary cyber physical systems (eCPS). In the course of realization of eCPS, it is necessary to enhance the models used for representation of components, objects and events etc. with sensorial data as collected from environment and formulating methodology for reasoning over such data by considering contextual data, more importantly temporal dynamics. The behavioural modelling paradigm and situational awareness will be presented with formal model that will be utilized for the modelling of the CPS. At the same time knowledge acquisition, and knowledge management paradigms by considering the learning systems from AI will be formulated within the scope of this research work. And, finally as stated in the motivation, the vision is to present a reference architecture and technical solution that will contribute in the cognition and configuration level of 5C architecture of CPS.

That is per se an interesting advancement towards deploying smart cyber physical systems, which will inherit natural behaviour and can pave path towards more robust CPS, may be one day *with immune system very similar to human body*.

1.4. RESEARCH AREAS – AN OVERVIEW

The scientific domains of influence for the PhD work is of course broad and involves multiple areas from pure science: natural evolution, biological organisms etc.; artificial

intelligence: evolutionary computation, evolutionary algorithms, formal modelling and core information technology: knowledge engineering, software engineering, system architecture and system modelling.

Even though these different domains play and important role in providing the background knowledge for the research work (which will be explained in detail in section II). Figure 1-4 shows the important research domains forming the base for this PhD work. In this sub-section we will present four important areas where we have envisioned providing concrete contributions.

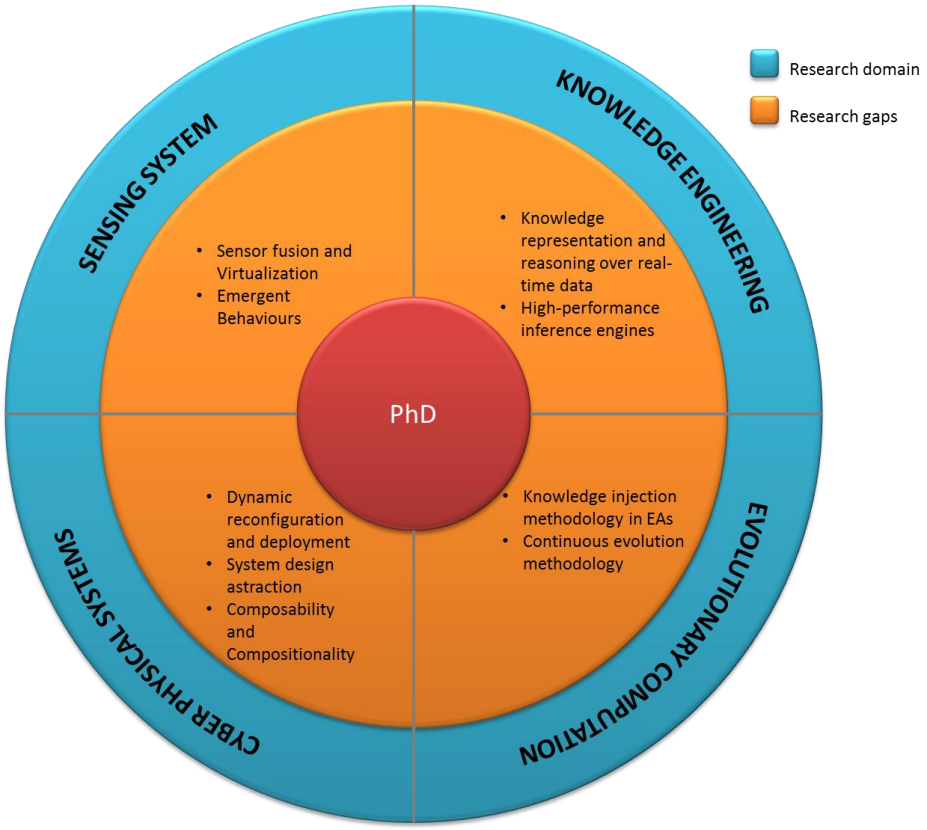


Figure 1-4 Important research domains and research gaps addressed in current research

1.4.1. *KNOWLEDGE ENGINEERING*

Knowledge engineering (KE) was defined in 1983 by *Edward Feigenbaum and Pamela McCorduck* as follows: KE is an engineering discipline that involves integrating knowledge into computer systems in order to solve complex problems normally requiring a high level of human expertise [20]. KE involves five steps i.e. Knowledge acquisition, Knowledge validation; Knowledge representation, Inferencing; Explanation and justification. Based on the principles of KE we can develop knowledge based system that uses artificial intelligence techniques in problem-solving processes to support human decision-making, learning, and action which is the research domain for this PhD work. In the scope of CPS, knowledge-based systems could be used for different purposes: fault

diagnosis that implies detection, cause analysis and repetitive problem recognition; complex control schemes; process and control performance monitoring and statistical process control; real time Quality Management (QM); control system validation [21].

1.4.1.1. THE GAP

Given the criticality of the CPS systems KE must address the issue of reliable methodology to meet the real-time constraint. At the same time, the systems produced through knowledge engineering methods must be able to re-use not only abstract ideas, but also implementation knowledge extraction and utilization from independent data producers. To do this, gaps in the domain of portability and interoperability must be addressed. The other important gap is in the knowledge representation of real-time scenarios which has Time-variant behaviour; Nonlinear, slow and irreversible process dynamics; infrequent data generation; constrained operation; Presence of disturbance effects etc. How can these factors be integrated in the knowledge extraction and representation phase? And the biggest gap in this domain is the development of high-performance inference engines that can guarantee response times.

1.4.1.2. FORESEEN CONTRIBUTION

A specific area for contribution is application of generic programming for knowledge extraction tasks. It is thus necessary to address a proper knowledge representation and reasoning methodology over real-time environmental data. The knowledge model should also consider emergence of relevant correlations while maintaining consistency of the overall knowledgebase. Thus, in general the expected contribution in this domain is novel methodology for knowledge engineering in real-time systems by considering the capability to model events, situations, context and reason over them, while maintaining temporal constraints.

1.4.2. *SENSING SYSTEMS*

Sensing system is capable of making decisions by using information captured through physical and virtual objects and providing added value information to enhance its global context awareness. In fact we have generalized the term sensing enterprise as defined by FInES Cluster [22]. The sensing system combines the concepts of sensors with mobile technology and distributed intelligence to perform analysis and decision-making, both in the real and digital worlds. This concept is a cornerstone for enterprise level CPS systems and is supported by the anticipation that sensors will become a commodity in future [23]. Note that sensing system will not only be dependent on physical sensors but also on

virtual sensors i.e. enterprise will not have physical access to the sensor but can have access to the observations of these sensors, which is also highlighted in [23].

1.4.2.1. THE GAP

Lately, the methodology for creating mash-up of sensorial data follow rigid approach of hard coded algorithm for data fusion from different sources i.e. the process is not standardized or not open source, and do not fit in enlarged models that accommodate large network of sensors dynamically. As, the diversity of device protocols and their properties rises, it requires the need for implementing integration solution. At the same time it will require the process for virtualization of physical resources so that it can be shared and utilized by different entities in the system ecology. The next research gap is development of data mining algorithms to estimate unobserved spectra of data within the data collected by the physical sensors.

1.4.2.2. FORESEEN CONTRIBUTION

Sensing from the physical world is an important aspect for this research work. Thus, the first important expected contribution is the development of generic solution for seamless integration of new devices into the existing system. The second contribution will be the solution for virtualization of physical devices. And the final expected contribution is towards creating algorithms that can predict and estimate missing values in a stream collected from physical/virtual sensor to help in predictive monitoring and analytics.

1.4.3. *CYBER PHYSICAL SYSTEMS*

CPSs have the potential to represent more than networking and information technology, information and knowledge being integrated into physical objects. By integrating perception, communication, learning, behaviour generation, reasoning into such systems a new generation of intelligent and autonomous systems may be developed [24]. CPS engineering principles includes design, specification, modelling, and analysis. And, engineering of CPS is complicated by a number of factors including heterogeneity, unreliable network communication, mobility and a tight coupling with the physical environment. At the same time, these characteristics introduce a level of uncertainty that is difficult to capture using traditional formal modelling techniques and thus requires appropriate formalism and reference architecture to form the basis for CPS engineering.

1.4.3.1. THE GAP

CPS research is still in its infancy and research is partitioned into isolated sub disciplines such as sensors, communications and networking, control theory, mathematics, software engineering, and computer science [25]. For example modelling formalisms are not complete and often represents either the cyber or the physical process well. At the same time engineering methodologies for CPS lack standardized abstractions and architectures that permit modular design. At the same time there are gaps in technical implementation of components that interact through a complex, coupled physical environment. And the bigger gap that the research community needs to fulfil is the need of hardware and software components that are highly dependable, reconfigurable, and in many applications, certifiable at the same time considering trustworthiness extended to the system level.

1.4.3.2. FORESEEN CONTRIBUTION

Methodology for modelling CPS is one important area that this research work will explore into aiming to provide considerable contribution. Work in modelling formalism to properly express both the cyber and physical domain of CPS is thus an expected contribution. The other foreseen contribution is building up technological framework that can be used for realization of scalable and robust CPS, along with the implementation of necessary generic modules necessary for building CPSs.

1.4.4. EVOLUTIONARY COMPUTATION

Evolutionary computation (EC) is inspired by biological evolution and is often used in computer science to solve global optimization by utilizing the principles of soft computing i.e. it is tolerant of imprecision, uncertainty, partial truth, and approximation. In general ECs are family of population-based trial and error problem solvers with a metaheuristic or stochastic optimization character. ECs have applied to solve various applications in the real-world domain to address problems such as production process planning, inventory system and supply chain network optimisation, task-based jobs assignment, mechanical/ship design tasks that involve runtime-intense simulations, data mining for the prediction of soil properties, automated tissue classification for MRI images, and database query optimisation etc. [26]. Considering the uncertain nature of the problems being addressed by CPS, ECs based problem solving technique can be an important research area.

1.4.4.1. THE GAP

One of the major gap in the ECs is to find solutions to industrial problems with development of decision-support systems that require continuous flow of data, predictive components, almost immediate recommendations for recovering from sudden changes, etc. [27]. These types of problems usually deal with many variables, nonlinear relationships, huge varieties of constraints (e.g. constraints in real-world settings often include 'if-then' conditions), business rules, many (usually conflicting) objectives – and all of these are set in a dynamic and noisy environment, which is highly observed in CPSs. At the same time utilization of domain knowledge for improving the results of ECs has not been explored thus requiring methodology for injecting knowledge into the evolutionary algorithms. Also, it is important to note that in the CPS domain, the optimization problems require recommendations for "the best" decision at the "moment". Another important gap is the optimization strategies for EC algorithms in terms of resources and time, which requires formulation of parallel genetic algorithms. One interesting approach is to adopt the Master-slave Model (MSM) which allows distribution of crossover and mutation operations, and in some cases fitness calculation, distributed over different processors. This allows utilization of the computing power of several processors or distributed computer systems to solve the problem [28].

1.4.4.2. FORESEEN CONTRIBUTION

In the scope of this research, the major contribution is made towards formulation of methodology for injecting knowledge in the evolutionary algorithms. At the same time this research work is focused on application of EC for solving multi-objective real world industrial problem by considering the effect of data collected from real-world with almost real-time frequency.

1.5. DISSERTATION OUTLINE

This dissertation is organized in four major sections. Each of the sections are as described below:

Section I: Introduction and Research Plan

This section builds up the foundation for the work performed within the scope of this PhD research which mainly includes all relevant information on the nature of the work performed and hereby reported. This section starts with an introductory ***Chapter 1*** that establishes the motivation and vision for conducting the present research work. It starts with the observations made by the author in the technological, industrial and academic paradigm to build up the vision that can be useful to take the state-of-art a step ahead. Based on the identified problem and stated motivation, most important research areas are

presented along with the gaps and aimed contributions in each domain. Chapter 1 ends with the outline of the overall document, to provide a good outline for the reader.

Chapter 2 in this section discusses the research methodology followed during the PhD research. Important content of this chapter is the formulated research questions and hypothesis. Also, some interesting rational on research questions is presented to provide deeper insight of the research work and the possible impact.

Section II: Literature Review

This section is the part where detailed studies on different domains of sciences ranging from natural theory of evolution to programming paradigms have been presented and is divided into two important chapters (***Chapter 3*** and ***Chapter 4***). This section is particularly important for understanding the background information considered necessary for the development of the research work. It starts with discussion on "Evolution" which includes both the natural theory of evolution (biology perspective) and evolutionary computation (computer science perspective). The comparison between these two domains helps to find similar and distinct paths to be undertaken for the research work. In the next sub-chapter study on CPS is provided for understanding the context and specificity of CPS.

This section also takes the discussion into complete computing domain and builds clear understating of evolutionary systems and points out necessary requirements to realize eCPS. Also, the discussion on programming paradigm is important to select the best suited programming language type for working with eCPS. This section ends with the summary of the literature review where it is provided the definition of some of the most important concepts that define the key-words of this research work. And, finally ***Chapter 5*** provides the synthesis of overall literature review with highlights on some important literatures that have form the base for the PhD work.

Section III: Theoretical and Technological Results

This section is the core of this dissertation and is divided into three chapters, providing the results of the PhD research work. Firstly more scientific or theoretical results are presented followed by technological results along with use-cases.

Chapter 6 provides the necessary theoretical and methodological foundation for realization of eCPS. The first half of the chapter provides formal model of evolutionary systems along with the modelling of behavioural and sensorial part of CPS. At the same time it includes methodology for domain knowledge injection in evolutionary algorithms, which is the foundation of all the technical work done afterwards. While the second half of the chapter is focused on methodological and technical foundation for design and implementation of CPS. This sub-chapter provides the methodological approach for realization of CPS and eventually eCPS. The technical methodology starts with reference architecture that will be used for all the technological developments of systems in the scope of this research work. The reference architecture accommodates all the

requirements that have been discussed in chapter 3. This chapter also provides discussion on technological blocks that will facilitate development of eCPS and the dynamic deployment framework for run-time environment of eCPS.

Chapter 7 presents the details on the technical implementation and industrial validation scenarios that have been developed in the scope of this work. This is an important chapter, because this chapter presents exploitable technological solutions that have been developed. Also, the work presented in this chapter is supported with use-cases from different industrial scenarios, which have been used to validate the working of the developed solution.

Section IV: Discussions and Prospective

This section presents the closing remark on the research and the dissertation document by analysing the results, establishing conclusions and opening perspectives for future research.

Chapter 8 revisits the previous chapters to build the connection between the problems that have been identified in the beginning of the document and results presented in following chapters. Also, quick analysis on the thesis outcomes to clarify some of the rational questions that have been raised provides interesting insight into the results.

Chapter 9 provides the concluding remark and some insights into the future work that can be undertaken to improve the state of evolutionary computation in CPS domain.

2. RESEARCH METHODOLOGY

A neat and orderly laboratory is unlikely. It is, after all, so much a place of false starts and multiple attempts.

- Isaac Asimov

In this chapter it is presented the research methodology that has been followed during the execution of this PhD research work. At the beginning of the chapter is presented the workflow with different steps that have been executed at different phases that has lead towards this dissertation. Following sub section i.e. section 2.1 presents the introduction to the research topic along with research questions that has been formulated to perform the research work. This section also, includes an interesting discussion on rational on research questions, that raises some issues which are not only addressed in this thesis, but also paves paths for future research. This chapter ends with stating the hypothesis to be validated by this thesis which is supported by the brief introduction to the approach that has been followed to achieve research results. The contents of this chapter not only direct the literature review in the following section but also the overall PhD research.

The adopted methodology is based on a research strategy that follows a workflow as mentioned in the course [29] and was executed following the classical phases, in its execution scheduling.

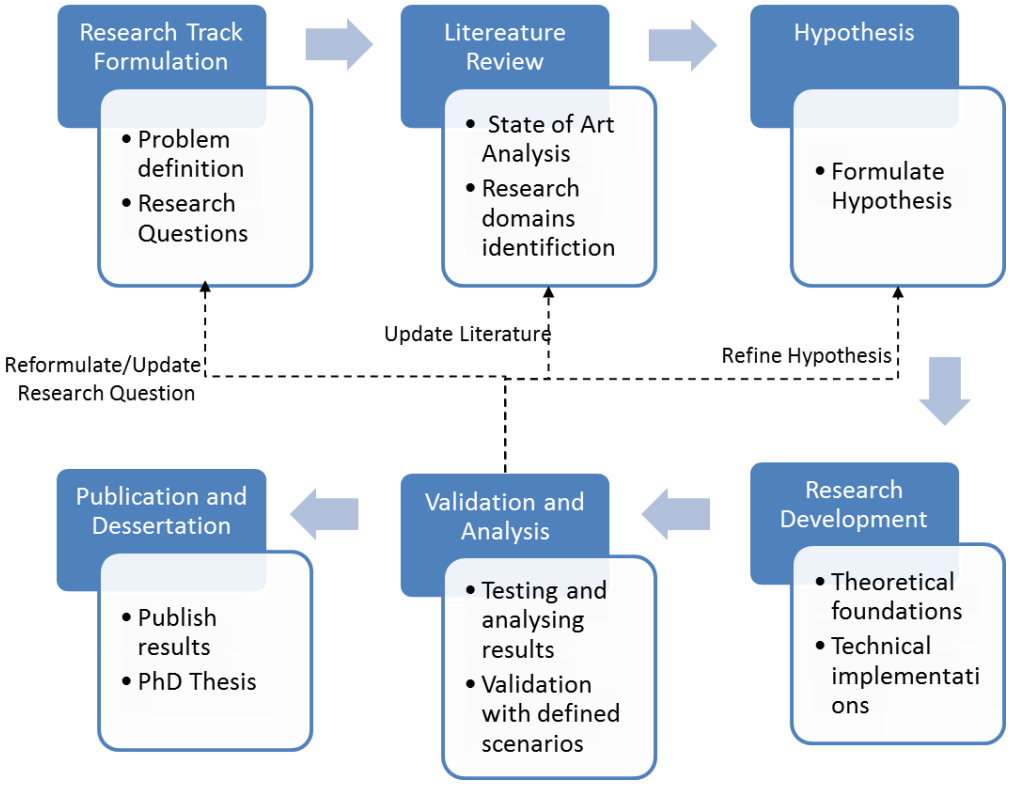


Figure 2-1 Research methodology

The important steps for the research methodology are as show in Figure 2-1, along with necessary dependencies and actions between some of the steps. Each of the steps of the research methodology is as explained below:

1) **Research Track Formulation:**

This step is dedicated towards defining the research track and framing the problem that needs to be studies and solved during the period of doctorate research. One important task in this step is formulation of research question and often comes from the thought: "What we have now is not quite right/good enough – we can do better…" [26]. This sentence can be used to capture the essence of this on-going research work. For instance in the scope of this research work the first thought that came to the mind was: Evolutionary computation has not been applied to complex systems, so what do we have to improve the understanding of complex systems like CPS, and apply the theory of evolution towards realization of solution that can help towards achieving evolutionary systems.

Also, note that the question emerges from existing knowledge in different domains, for instance "theory of evolution" and from the recommended research areas of the PhD program for instance "Industrial Information Systems" in the scope of this research work. The research question give the track for the research work and needs to addressable by a research development that will lead to conclusions on its feasibility or its proven value. The research question and sub-questions are presented in section 2.1.2.

2) Literature Review:

After the formulation of the research domain and understanding of the problem, it is necessary to study the literatures for understanding of the state of art in the domains related to the problem. From the observations of problem and the targeted scientific environment, this step allows for the selection of important literatures covering the topics considered relevant for the execution of the proposed research work. This step thus, helps in the clear identification of scientific and technologic domain and covers up the study on previous researches. This step is thus important or identification of the gaps in the state of art and defines distinguished contribution from this research. Section II provides literature review on different areas considered necessary for the research plan.

3) Hypothesis:

The results from step 1 and step 2 provide the necessary inputs for formulation of hypothesis, which will be a supposition or proposition made on the basis of limited evidence (problem definition and literature review) and acts as a starting point for further investigation. The research hypothesis is a paring down of the problem into something testable and falsifiable. It is necessary to generate a realistic and testable hypothesis around which experimentation and research can be performed. The hypothesis for this research work is mentioned in section 2.2 and is formulated by taking into account that a hypothesis should be; simple and conceptually clear, capable of verification, related to the existing body of knowledge from several confluent areas.

4) **Research Development:**

This is one of the longest periods during the research development of the PhD work. During the course of this step the theoretical foundations are to be built to solve the research questions raised in step 1 leading towards the validation of hypothesis. For instance in the scope of this work, it's necessary to formulate the theoretical foundation related to evolutionary computation, the way to formally establish the theory, CPS system theory, behavioural modelling of system that works in close contact with the environment etc. Thus, built theoretical foundation will be the base for providing the technical implementation which can then be used for experimentation. In this phase it's also important to define experimentation plan that will allow the validation of the hypothesis towards the confirmation of thesis. In designing the testing setup it is necessary to identify the variables to be manipulated and measured, what outcomes will be measurable and, in summary, to develop a method to reach the necessary validation. The technical implementation to be performed also needs to be tested with some industrial cases if possible and via simulations where industrial case is not applicable.

5) **Validation and analysis:**

The result from the previous step needs to be validated and analysed in order to formulate thesis from hypothesis. The theoretical foundation needs to the validated via supporting technical validation. The technical implementation can be validated by analysis of the data collected at different steps of execution of the technical solution. The data collected at each step of execution need to be evaluated qualitatively and qualitatively using valid technics (e.g. graphical analysis, statistics, etc.).

The analysis of the results to interpret the data is an important task in this step. The main objective of this task is testing the hypothesis using the proposed design experiment aimed to confirm or refute that same hypothesis. The analysis of collected data allowed the verification of the proposed hypothesis, proving that it is possible to improve knowledge management by collecting physiological data. Note that in this step if the followed path leads to the absence of conclusions or if the hypothesis is proven wrong, it could be better to choose another problem to address or view the problem from another angle so that, with some literature review would be possible to successfully validate another hypothesis.

6) **Publication and Dissertation:**

The results obtained during the research activities have been published, and will continue to be disseminated, for the benefit of the scientific community, and also to allow validation from peers.

2.1. RESEARCH QUESTION

It has been discussed in the previous section; a research question is the fundamental core of a research activity. Research question helps to establish the focus of the study,

formulate research development strategy and, and guides all stages of inquiry, analysis, and reporting. In the following subsections we will formulate the research question and make some rational discussions some of the important aspects of the research domain.

2.1.1. *Research Topic*

The topic chosen and reflected in the dissertation title is **"Self-Evolutionary Cyber Physical Systems"** which is a sub-topic of the wider domain: "**Cyber Physical Systems**". The rational for the selected field of research has been explained before in different sections analysing various facets of the research work.

2.1.2. *Research Questions and Scientific Challenges*

Research Question helps in identification and definition of the research area and frames the problem to be addressed by the research work. The research question is formulated based on the motivation scenario and the major research challenges as explained in section 1. The formulated research question will raise new research topics that will reshape the formulated hypothesis. This process aims the construct of new scientific knowledge as stated in the objectives of the reported research work

Question – "How can the principles of machine learning and automated reasoning contribute for realization of self-evolutionary Cyber Physical Systems?"

To assure the research focus and targeted results, the major question is split threefold according to tree major steps towards solutions i.e. "***Detection of a problem***", "***Response to the problem***" and "***Impact of the solution***":

- ***The detection of a problem***

SQ1 - "How can the dynamics of physical world be monitored and captured to detect changes and/or failures in CPS?"

- ***The response to the problem***

SQ2 - "Which methodology and tools need to be developed to support modelling, implementation and integration of dynamics in physical processes into the computational systems to achieve self-evolutionary CPS?"

- ***The impact of the solution***

SQ3 - "To what extent can the self-evolutionary scheme applied in CPS affects the overall performance of the system?"

2.1.3. *Rational to Research Questions*

In this sub-section, we have provided an interesting discussion on rational on overall research work along with the ones related to the research questions stated in the sub-sections before. The main objective is to have an insight into different issues and perspective that will be within the scope of this research work. It is important to note that not all the questions raised below are addressed in the thesis, but it definitely paves path for future research in the same domain.

2.1.3.1. Rational 1: System Dynamics: Unreliable Input vs. Reliable Systems

CPSs deal with physical world and data is collected from devices like sensors work in a more hash and open environment than those of traditional networking systems, making inaccurate, redundant data as well as attacks a common phenomenon, rather than exceptional. Additionally, the most fundamental difference between CPS and traditional systems is that time plays an essential role in CPS. The real physical world presents a large seamless concurrent unit, bringing more uncertainties to systems than a real-time system can handle. So, unreliable data would be inevitable and an impediment to the progress towards evolutionary CPS design. [30] lists the whole spectrum of data management in sensor networks, suggesting two relating subfields in manipulating sensor data: statistical modelling, data uncertainty handling.

This opens up a research challenge to how to define methodologies and algorithms for the detection of changes and/or failures in the domain of uncertainties. *At the same time we have to consider what aspects of the environment should the self-evolving system monitor? What exactly should the system do if it detects a less than optimal pattern in the environment? How should the system deal with the dynamics that are not predictable? And how should the system self-evolve in order to prevent the fatal errors caused by the system dynamics?*

2.1.3.2. RATIONAL 2: SYSTEM ENGINEERING: EVOLUTION VS. AGILITY

Robustness and efficiency used to be two key issues in system design, but is it sufficient in the context of CPS? Future CPS should feature as environment-aware, which differentiates from existing complication trade-offs because of the infinite complexity of the physical world compared to finite algorithms and system resources. Besides, a well-known characteristic of the physical world, but often ignored currently leads systems to be fragile and vulnerable. For instance, it is relatively doable to optimize part of CPS, which might trigger potential chaos to the system as a whole, like the butterfly effect--*when every corner of the system adjusts its rules, how does it react to the whole picture?*

It is thus important to define methodology for implementation and integration of the physical and computational world so that the changes in the system can be achieved in agile fashion. *And how should the system self-evolve in order to prevent the fall in performance or increase in operation cost or changes in other factors within the acceptable limits?* Also it raises the question - *how can the system identify the differences between critical changes and non-critical changes and choose the correct path for evolution?* Also from dynamic observation on the changes in the requirements based on the un-predicted scenario that occurs in the physical process needs to be incorporated in the methodology of system engineering. *Can the system requirement artefacts be turned into run-time objects?*

2.1.3.3. RATIONAL 3: EVOLUTIONARY METHODOLOGY: SYSTEM AUTONOMY VS HUMAN INTERFERENCE

CPS design becomes more and more challenging than traditional computer system design in that CPS places accountability at a much higher level than traditional systems, because it is a miss-critical and life-threatening system. Evolutionary methods to be developed for such systems raises an important question on -*how and/or when human actions should be taken into the evolutionary actions?* At the same time it is also important to define a mechanism to revert evolutionary path the system has undergone by human intervention. So, the question over the degree of flexibility for autonomy of a CPS system has to be handled well in the scope of eCPS.

2.2. HYPOTHESIS AND APPROACH

Based on the discussions in the previous sections, research questions and discussion on rational of the challenges, one can estimate that:

Hypothesis- If a learning mechanism is developed and applied during the control loops of system behaviour then evolutionary actions can be extracted and incorporated to achieve self-evolutionary Cyber Physical System.

The above statement is therefore the adopted hypothesis that will be implemented, tested and challenged during the PhD work. The main key words to be extracted from the hypothesis are ***learning*** and ***behaviour***. These keywords have guided this research work leading towards this thesis.

The hypothesis can be detailed such that for defining learning mechanism in the domain of CPS will require taking into consideration of the semantics specific to the CPS. In order to aim at developing work towards the proposed hypothesis, and from the early analysis of the problems, we observed the next set of tools or developments.

1. Understanding of the theory of evolution and evolutionary computation and formulate the theoretical foundation that can be used for defining evolutionary cyber physical systems.
2. Behavioural modelling formalism that can be used to model the functional behaviour of systems working in un-controlled environment and model the situations to infer necessary actions from events.
3. Sensorial system modelling i.e. effective formalism to model system composed of number of sensors and actuators and the way how they can be integrated to build more complex systems.
4. Generic design and modelling methodology for (e)CPS along with reference technical architecture for realization of evolutionary systems. It will include the instantiation of modules for sensing, monitoring, adapting, learning and eventually evolving.
5. With the proposed approach a CPS will be enriched with effective monitoring and learning features with knowledge acquisition at various steps that can be used to compute evolutionary cycles for CPS.

Our primary goal is to facilitate better understanding and representation evolutionary computation and CPS and eventually trace evolutionary paths by enabling different types of information representation and extraction based on sensorial systems.

On a quick glimpse on some of the background knowledge, shows promising sign successful realization of the above steps. Learning methodology can be developed by following the Artificial Neural Networks (ANN) models of Artificial Intelligence (AI) or Knowledge Representation and Reasoning (KRR) theories of Computational Logics. The path to be chosen for this research work is yet to be finalized but it is important to define well-formed semantics mapped across various layers of CPS systems. KRR make use of formal languages with syntax, semantics and inferences to define reasoning techniques, which can be a path to be adopted for this research work. Another interesting domain which can provide strong base for this research work is multi-agent systems which allow formulating problems by defining a group of autonomous, interacting entities sharing a common environment, in which they perceive with sensors and upon which they act with actuators. The target of the research work is thus towards CPS which are capable of

learning based on its environment and behavioural functionalities so that the resulting systems are adaptive (responding to changing conditions) and predictive (anticipating changes in the physical processes) based on the acquired knowledge at run-time.

SECTION II: LITERATURE REVIEW

In the previous section, we have built the backbone of the research path that has been guided by the highly dynamic nature of CPS. One of the important base and guidance for this research work has come from the "Theory of Evolution" in natural science. This section includes the study on current knowledge, as well as theoretical, methodological and technical contributions in particular topics that have been followed for the research work. This section is particularly important to see the big picture of the research problem and understand some of the existing research results and activities that have acted as the starting point of this dissertation. It is to be noted that the research questions have been presented in the preceding sections of this dissertation, the studies made in the scope of this section has contributed the formulation of research questions and hypothesis.

This section is composed of four chapters divided into well classified research domains and ends with a summary on the merge of three important areas i.e. Evolutionary Biology, Computation and Cyber Physical Systems. First Chapter provides the insight into the research domain of Evolution, with insights into theory of evolution, evolutionary computation and comparative study on evolution in biological science and computer science. The following chapter provides the discussion one core domain of this research work i.e. Cyber Physical Systems. In chapter 5 is presented the studies on evolutionary systems, which is one of the core challenge addressed by this thesis. Since, CPSs are class of complex systems, it is very important to follow the best programming paradigm for the technical implementation. The studies made in this direction are presented in chapter 6 Programming landscape.

*Some of the most important summaries of this section are: "... **evolution in systems can be systematically studied, changes can be monitored and the system can learn through the changes that the systems undergo**... (c.f. 3.3) " ; "... **evolutionary transitions are usually gradual, i.e., new species evolve from pre-existing varieties by slow processes and maintain at each stage their specific adaptation** ... (c.f. 3.1)"; "...**the cyber and the physical subsystems coexist in time … feedback loop between physical processes and computations encompasses sensors, actuators, physical dynamics, computation, software scheduling, and networks**... (c.f. 4) " and "... **programming languages class to be used should be modular with dynamically linking and network centric**... (c.f. 4.3)";*

3. EVOLUTION

The affinities of all the beings of the same class have sometimes been represented by a great tree. I believe this simile largely speaks the truth. The green and budding twigs may represent existing species; and those produced during each former year may represent the long succession of extinct species.

-Charles Darwin

This chapter makes the introduction to aspects of evolution which has been studied by various scientists for an understanding about change in the heritable characteristics of biological populations over successive generations. In the first section of this chapter, we present the introduction on the theory of evolution from natural science followed by injecting computing paradigm via discussions on evolutionary computation. In the following sections we make comparative study on evolution in biology and computer science. The chapter ends with discussion on how theories of evolution can be used for problem solving. By the end of this chapter, reader is expected to be familiar with overall concepts of evolutionary theory and its application in computing.

Evolution is defined as the change through time as species become modified and diverge to produce multiple descendant species, having different often better functional behaviours. Evolution and natural selection are often conflated, but evolution is the historical occurrence of change, and natural selection is one mechanism—in most cases the most important—that can cause evolution. The two main pillars of our knowledge of evolution come from knowledge of the ***historical record of evolutionary change, deduced directly from the fossil record and inferred from examination of phylogeny***, and from study of the ***process of evolutionary change, particularly the effect of natural selection***. It is now apparent that when selection is strong, evolution can proceed considerably more rapidly than was generally envisioned by Darwin [31]. As a result, scientists are realizing that it is possible to conduct evolutionary experiments in real time.

In the context of computing, *L.G Valiant* provides the definition of evolvablity is a restricted case of Probably Approximately Correct (PAC) learnability [32]. PAC Learning Model that was introduced by *L.G Valiant*, of the Harvard University, in a seminal paper [33] on Computational Learning Theory way back in 1984. The PAC model belongs to that class of learning models which is characterized by learning from examples. Thus, considering evolvability as a case of PAC learnability offers a unifying framework for the fields of evolution and cognition. The behaviour of a biological organism is clearly affected both by the results of evolution and those of learning by the individual. Distinguishing between the effects of nature and nurture on behaviour has proved problematic, and it will perhaps help to have a unifying viewpoint on them.

One important aspect for consideration in evolution is- assume that the updates depend only on the aggregate performance of the competing hypotheses on a distribution of examples or experiences, and in no additional way on the syntactic descriptions of the examples.

3.1. THEORY OF EVOLUTION

The theory of evolution, formalized by Charles Darwin, is as much theory as is the theory of gravity, or the theory of relativity. Unlike theories of physics, biological theories, and especially evolution, have been argued long and hard in socio-political arenas. However, evolution is the binding force of all biological research. The history of evolution in general is useful to the researchers from different domains. At the same time evolution remains a relatively under researched topic within the science education community [34].

In biology, the "science of the living world," both past and present [35], the situation is very different. The organisms biologists study, which are typically randomly drawn from populations, manifest astonishing variation as a consequence of genetic recombination and random genomic changes. However, there are limits to biological variation and these literally shape evolutionary history. No population is ever capable of generating all possible theoretical genomic variants, in part because sexual genetic recombination is random and because the existence of any particular population is finite. Therefore, biological variation, which provides the "raw material" for evolutionary change, is confined by random events [36]. Nevertheless, non-random processes also shape evolution. The "struggle for existence" among the offspring of each generation eliminates genomic variants that are less adapted to their environment. Those that survive pass their genetic information on to the next generation. In this way, evolution is the summation of random events (e.g., mutation and sexual recombination) and natural selection, which is largely non-random. The major propositions from the Darwin's theory of evolution are presented in Table 3-1 as taken from [39].

Table 3-1 Principal propositions of Darwin's theory, extracted from the Origin of Species

1. Supernatural acts of the Creator are incompatible with empirical facts of nature
2. All life evolved from one or few simple kinds of organisms
3. Species evolve from pre-existing varieties by means of natural selection
4. The birth of a species is gradual and of long duration
5. Higher taxa (genera, families etc.) evolve by the same mechanisms as those responsible for the origin of species
6. The greater the similarities among taxa, the more closely they are related evolutionarily and the shorter their divergence time from a last common ancestor
7. Extinction is primarily the result of interspecific competition
8. The geological record is incomplete: the absence of transitional forms between species and higher taxa is due to gaps in our current knowledge

Ernst Mayr provides interesting summary for understanding the evolutionary process – "Gradual evolution can be explained in terms of small genetic changes ('mutations') and

recombination, and the ordering of this genetic variation by natural selection" [37]. Some other important considerations for understanding the process of evolution are as summarized below:

- Natural selection is the most important force that shapes the course of phenotypic evolution. In changing environments, directional selection is of special importance, because it causes a shift in the population mean towards a novel phenotype that is better adapted to altered environmental conditions.
- The evolutionary transitions are usually gradual, i.e., new species evolve from pre-existing varieties by slow processes and maintain at each stage their specific adaptation.
- Macroevolution (i.e., phylogenetic developments above the species level or the occurrence of higher taxa) is a gradual step-by-step-process that is nothing but an extrapolation of microevolution (origin of races, varieties, and species).

These points are important aspect for guiding research in the scope of this PhD thesis. It is also important to note that in some important literatures like [38,39], [40,41] researchers have pointed out that the patterns and controlling forces of evolution are much more varied than were postulated by the pioneers of evolutionary biology (Darwin, Wallace, Weismann) and the "architects" of the synthetic theory (Dobzhansky, Mayr, Huxley and others). The expansion of the modern picture of the mechanisms of evolution is discussed in article [36], which deals with some very interesting topics as shown in Figure 3-1

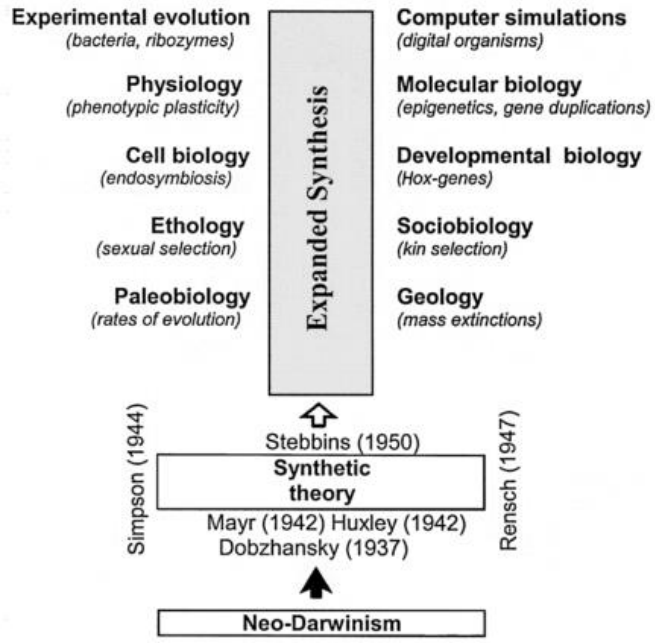


Figure 3-1 Scheme illustrating the expansion of the synthetic theory of biological evolution by integration of ten additional scientific disciplines

In any organism no single function is more important than any other, because environmental factors influencing one or more parts of the phenotype indirectly or

directly affect the whole organism. Importantly, different biological tasks have different phenotypic requirements and some tasks have antagonistic design requirements. An additional insight from these simulations is that natural selection cannot create a "perfect" organism, because optimization results in organisms that perform all of their functional obligations reasonably well simultaneously but not perfectly in terms of each individual task [36]. On, the whole theory of evolution of based on the idea - repeated formation of new species (speciation), change within species (anagenesis), and loss of species (extinction) throughout the evolutionary process that are demonstrated by shared sets of morphological and biochemical traits, including shared DNA sequences [42]. Hence, in nature, evolution is mostly determined by natural selection or different individuals competing for resources in the environment- ***individuals that are better are more likely to survive and propagate***. Reproduction allows some exchange and re-ordering of chromosomes, producing offspring that contain a combination of information from each parent. This is the recombination operation, which is often referred to as crossover because of the way strands of chromosomes cross over during the exchange and diversity in the population is achieved by mutation.

Based on the discussions presented in this section, it's quite clear that a number of scientific disciplines contribute in the current understanding of evolutionary theory. In the scope of this thesis, we are concerned with the principles of *digital organisms*. Digital organisms are self-replicating computer programs that live in a controlled environment. Digital organisms explicitly create a copy of their own genome to reproduce, and no particular genomic sequence is designated as the target or optimal sequence. Selection occurs because the environment in which the digital organisms live is space limited, that is, with the birth of a new organism an older one (typically chosen randomly) is removed from the population. Therefore, those organisms that produce more offspring replace less efficient replicators over time. Recent developments in computer technology and mathematical principles have provided the tools to model organismic evolution. The approach is to *simulate all conceivable phenotypic variants for a particular lineage or grade of organic organization* (i.e., to construct a "morphospace") and to quantify the performance of each of these variants in terms of one or more biological functions believed to influence relative fitness, such as visual acuity in animals or photosynthesis in plants (i.e., to *generate a "fitness landscape"*). For example, computer models have been used to mimic the early evolution of ancient vascular plants (tracheophytes) [43,44]. These models can be useful for modelling more *complex organism and/or digital organism*.

3.2. EVOLUTIONARY COMPUTATION

The field of evolutionary computation has many founders and many names- a concise summary of the origins of evolutionary computation can be found in [45]. Evolutionary computing is the collective name for a range of problem-solving techniques based on principles of biological evolution, such as natural selection and genetic inheritance.

Evolutionary Computation is the result of efforts researchers with the idea of mimicking mechanisms of biological evolution in order to develop powerful algorithms for problems of adaptation and optimization. Since many optimal structures like the shape of birds' wings or the branching structure of blood vessels have emerged through biological evolution, the idea to utilize the underlying mechanism for the solution of optimization problems has motivated a considerable amount of research, resulting in several approaches that have proven their effectiveness and robustness in a variety of applications (refer [46] for brief bibliography and history).

If evolution merely performed a random search, it would require exponential time, much too long to explain the complexity of existing biological structures and also for the case of complex systems. In this regard, Darwin suggested selection as the critical controlling principle beyond variation. He also observed that the supposition that the eye could evolve would be " ⋯ absurd in the highest possible degree" were it not for the fact that eyes "vary ever so slightly" and might therefore evolve over time by selection [47]. In other words, a necessary condition for evolution to a specific target is the existence of an evolutionary path towards it consisting of small steps. In particular***, in a defined quantitative sense, selection for a given beneficial behaviour can provably support the evolution of certain specific classes of mechanisms***, and provably not support that of certain other classes [32].

Evolutionary computation (EC) [48] is often grouped under evolutionary algorithms (EA) and in general is simulate the evolution of individual structures via processes of selection, mutation, and reproduction. This flow is depicted in Figure 3-2.

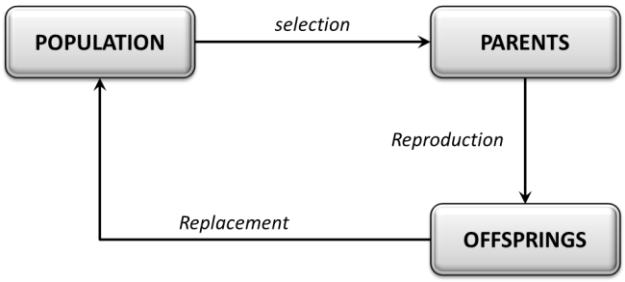


Figure 3-2 Generic Flow chart of an evolutionary algorithm

EC or EAs is composed of different domains such as : genetic algorithms [49], evolution strategies [50], [51], evolutionary programming [52] and genetic programming [53]. The processes depend on the perceived performance of the individual structures as defined by the problem. A population of candidate solutions (for the optimization task to be solved) is initialized. New solutions are created by applying reproduction operators (mutation and/or crossover). The fitness (how good the solutions are) of the resulting solutions are evaluated and suitable selection strategy is then applied to determine which solutions will be maintained into the next generation.

Evolutionary algorithms are ubiquitous nowadays, having been successfully applied to numerous problems from different domains, including optimization, automatic

programming, machine learning, operations research, bioinformatics, and social systems. In many cases the mathematical function, which describes the problem is not known and the values at certain parameters are obtained from simulations. In contrast to many other optimization techniques an important advantage of evolutionary algorithms is they can cope with multi-modal functions.

3.2.1. *EVOLUTIONARY COMPUTATION MODELS*

As discussed in previous section evolutionary computation is based on natural selection and genetics and a variety of evolutionary computational models have been proposed. There are different variations of the evolutionary computation models (refer [54] for detailed discussions) But, common to all is the concept of simulating evolution of individual structures using genetic operators such as selection, mutation, and reproduction. The operators used in evolutionary computation, is only a simplistic subset of biological processes. And, the fitness of each individual in the environment affects it's capabilities of surviving, or transferring its genes to succeeding generations

Table 3-2 The basic evolutionary algorithm

t:=0;	{initialize time}
InitPopulation(P,t);	{initialize random population of individuals}
EvaluateFitness(P,t);	{evaluate fitness of all initial individuals in the population}
while not terminate(P,t) do	{test for termination criterion, e.g. # of generations, satisfactory fitness, etc.}
begin	
t:=t+1;	{increase time}
SelectParents(P,Ps);	{select subpopulation for reproduction}
Recombine(Ps);	{recombine the genes of selected parents}
Mutate(Ps);	{mutate (perturb randomly) the mated population}
EvaluateFitness(Ps,t);	{evaluate new fitness}
Survive(P,Ps);	{select the survivors using actual finesses}
end;	

Based on the basic algorithm mentioned above, there are three main evolutionary computation models viz.: ***Genetic Algorithms, Evolutionary Programming, Evolutionary Strategies*** *and* ***Genetic Programming***, which are also depicted in Figure 3-3.

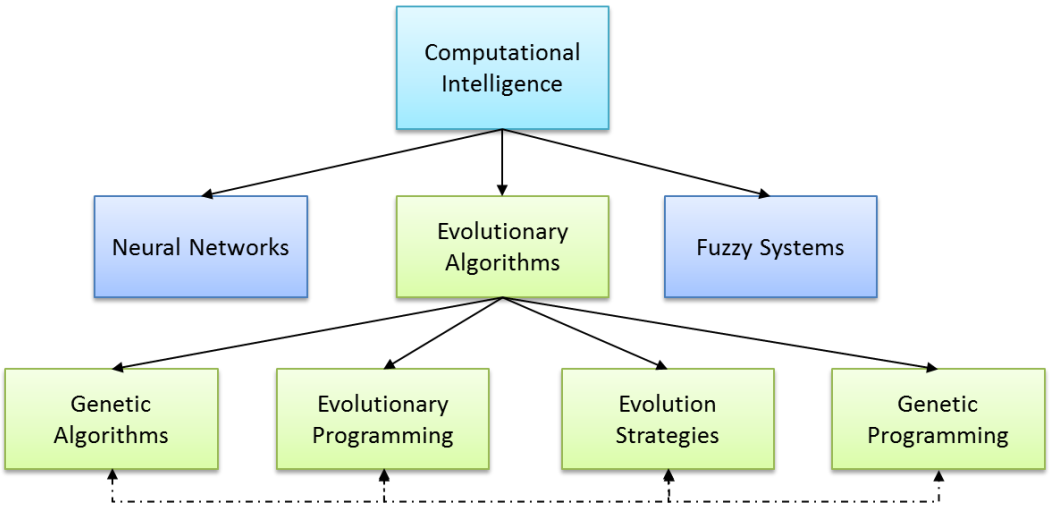


Figure 3-3 Branches of evolutionary computation

Also note that these models are closely related and differ mainly in the way the individuals are encoded and application of the genetic operator. These are discussed in the following subsections.

Genetic Algorithms

Genetic algorithms (GAs) are computer programs that mimic the processes of biological evolution in order to solve problems and to model evolutionary systems [55]. GA has the following components:

- a mechanism to encode solutions to problems as strings,
- a population of solutions represented as strings,
- a problem dependent fitness function,
- a selection mechanism,
- crossover and mutation operators.

Encoding mechanism is problem dependent and is the methodology to represent optimization problem strings. Generally, the resultant mapping is a fixed length binary string. It is possible to represent both discrete and continuous variables. However, in the case of real valued variables, the encoding is a two stage process. First, the real variable is mapped linearly to an integer defined in a specified range, and then this integer is encoded using a fixed number of binary bits. The binary codes of all variables are then concatenated to obtain a binary string. Authors, provide a good discussion on different encoding schemes used in GA[56] and also an interesting application scenario in [57].

Fitness function is used to evaluate the problem encoded string by using the normalized form of the objective function to be optimized. This fitness function has values in the range [0, 1] which are the fitness of the individuals in the population. Using the fitness of the string the selection mechanism evaluates them. Selection is based on the survival of the fittest mechanism of nature. Individuals who are more fit in some sense defined by the

problem survive whereas weaker ones perish. This operator has several different implementations. A fitter individual is allowed to have a higher number of offspring leading to an increased probability of surviving in the subsequent generations. In the proportionate selection scheme, a string with fitness value fs is allocated fs/fa offspring, where fa is the average fitness value of the population.

In nature, the encoding of genetic information admits asexual reproduction (such as by budding). This process typically results in offspring that are genetically identical to the parent, whereas sexual reproduction allows the creation of genetically radically different offspring that are still of the same species. Crossover is generally implemented as follows- pairs of strings are picked at random from the population to be subjected to crossover. Single point crossover is the simplest approach. Here, if the length of the strings is >0, then a crossover point which can have values in the range 1 to -1 is randomly chosen. The fragments of the two strings beyond this crossover point are exchanged to form two new strings if a randomly generated number is greater than $\boldsymbol{p_c}$ -the *crossover rate* which is a parameter of the GA. Crossover mechanisms such as two-point, multi-point, and uniform have been proposed as improvements to the single-point crossover technique (ref [58] for more details). While, mutation of a bit from the encoded string is carried out by changing a 0 to 1 or vice versa. Similar to $\boldsymbol{p_c}$ which controls the probability of crossover, another parameter, $\boldsymbol{p_m}$ -the *mutation rate* gives the probability that a bit will be altered. The mutation of a bit does not affect the probability of mutation of other bits.

And, the final consideration in GA is the generational cycle which consists of repeated application of selection and the genetic operators to the population. Typically a population size of 30 to 200 is used. Crossover rates are chosen in the interval [0.5, 1.0] whereas mutation rates in the interval [0.001, 0.05]. These parameters are called the control parameters of the GA and are specified before the execution starts. The generational cycle is terminated using a stopping criterion such as: reaching a fixed number of iterations; evolving a string with a high fitness value; creation of a certain degree of homogeneity within the population.

Evolutionary programming

Evolutionary programming (EP) [59] is a stochastic optimization strategy similar to genetic algorithms, but here both genome like representations and crossover operator is not used. Like GA, the EP technique is useful for optimization problems when other techniques like gradient descent or direct, analytical discovery fail. Combinatorial and real-valued function optimization in which the optimization surface possesses many locally optimal solutions, are well-suited for the EP technique.

As mentioned in the previous section, the typical GA approach involves encoding the problem solutions as a binary string. In the EP approach, however, the representation follows from the problem. For example, a neural network can be represented in the same

manner as it is implemented since the mutation operation does not demand a linear encoding.

In case of EP the mutation operation simply changes aspects of the solution according to a statistical distribution which weights minor variations in offspring as highly probable and substantial variations as increasingly unlikely as the global optimum is approached. There is a certain tautology here: if the global optimum is not already known, how can the spread of the mutation operation are damped as the solutions approach it? Several techniques have been proposed and implemented which address this difficulty, the most widely studied being the "Meta-Evolutionary" technique in which the variance of the mutation distribution is subject to mutation by a fixed variance mutation operator and evolves along with the solution [60].

Evolutionary Strategies

Evolution Strategies (ES) [61] sometimes also referred to as Evolutionary Strategies, are search paradigms inspired by the principles of biological evolution. They belong to the family of evolutionary algorithms that address optimization problems by implementing a repeated process of (small) stochastic variations followed by selection: in each generation (or iteration), new off- spring (or candidate solutions) are generated from their parents (candidate solutions already visited), their fitness is evaluated, and the better offspring are selected to become the parents for the next generation. In a two-membered or (1+1) ES, one parent generates one offspring per generation by applying normally distributed mutations, i.e. smaller steps occur more likely than big ones, until a child performs better than its ancestor and takes its place. Because of this simple structure, theoretical results for step size control and convergence velocity could be derived. the ratio between successful and all mutations should come to 1/5: the so-called 1/5 success rule was discovered.

A single individual of the ES population consists of the following genotype representing a point in the search space. *Object variables*: Real-valued x_i has to be tuned by crossover and mutation such that an objective function reaches its global optimum. *Strategy variables*: Real-valued i or mean stepsizes determine the mutability of the x_i. They represent the standard deviation of a $(0, i)$ gaussian distribution being added to each x_i as an undirected mutation. With an "expectancy value" of 0 the parents will produce offspring similar to themselves on the average. In order to make a doubling and a halving of a stepsize equally probable, the i mutate log-normally, distributed from generation to generation. These stepsizes hide the internal model the population has made of its environment, i.e. a self-adaptation of the stepsizes has replaced the exogenous control of the (1+1) ES.

This concept is successful because selection sooner or later prefers those individuals having built a good model of the objective function, thus producing better offspring. Hence, learning takes place on two levels: (1) at the genotype, i.e. the object and strategy variable level and (2) at the phenotype level, i.e. the fitness level.

Depending on an individual's x_i, the resulting objective function value $f(x)$, where x denotes the vector of objective variables, serves as the phenotype (fitness) in the selection step. In a plus strategy, the best of all (+) individuals become the parents of the next generation. The second scheme is more realistic and therefore more successful, since no individual may survive forever, which could at least theoretically occur if the plus variant is used.

Genetic Programming

Genetic programming (GP) is the extension of the genetic model of learning into the space of programs [62], [63]. Thus, in GP the objects that constitute the population are not fixed-length character strings that encode possible solutions to the problem at hand, they are programs that, when executed, "are" the candidate solutions to the problem. In GP we evolve a population of computer programs. That is, generation by generation, GP stochastically transforms populations of programs into new, hopefully better, populations of programs. The structures that are being adapted in GP are hierarchically structured computer programs whose size, shape, and complexity can dynamically change during the process. The set of such structures is the set of all possible composition of functions that can be composed recursively from the available set of n functions $F=\{f_1, f_2, ..., f_n\}$ and the available set of m terminals $T=\{a_1, a_2, ..., a_n\}$. Depending on the problem at hand, the functions may be basic arithmetic operations, Boolean operations or even iterative operators such as Do-Until, etc.

The basic steps in a GP system finds out how well a program works by running it, and then comparing its behaviour to some ideal scenario such as- how well a program predicts a time series or controls an industrial process. This comparison is quantified to give a numeric value called fitness. Those programs that do well are chosen to breed and produce new programs for the next generation. Almost all high level programming languages are suitable for expressing and evaluating the compositions of such types of functions. The search space for GP is the hyperspace of valid expressions that can be recursively created by composition of the available functions and terminals.

Initial random population is generated by selecting one of the functions from the set F at random to be the root of the tree. Whenever a point is labelled with a function which has k arguments, then k children are created randomly. If a child is a terminal, then the process is complete for that portion of the tree. If the child is a function, then the process continues. In GP during crossover the creation of a child program by combining randomly chose parts from two selected parent programs. While, for mutation- the creation of a new child program by randomly altering a randomly chosen part of a selected parent program.

3.2.2. *BIOLOGICAL EVOLUTION VERSUS COMPUTATIONAL EVOLUTION*

Beneficial traits resulting from random variation are favoured by natural selection, i.e. individuals with beneficial traits have better chances to survive, procreate and multiply, which may also be captured by the expression differential reproduction. In order to understand evolutionary algorithms, some basic notions are important, which will highlight the applicability of biological principles to computer science. Good resources for further details are: [64–66]. For an interesting comparison between the entities in the biological universe and computational universe follow Table 3-3, which is adapted from [67].

Table 3-3 Comparison between natural evolution and evolutionary algorithms

Natural evolution	Evolutionary algorithms
Allele	
Value of a gene	Value of information object
Chromosome	
DNA, protein and RNA sequence in cells (describes the construction plan and traits of an individual)	Sequence of information objects (describes the construction plan and traits of an individual)
Gene	
Part of a chromosome, as a fundamental unit of inherence, which determines a (partial) characteristic of an individual	Information object e.g. a bit, a character, a number etc. which determines a (partial) characteristic of an individual
Individual	
Living Organism	Solution Candidate
Fitness	
Observed quantity: a posteriori effect of selection and reproduction (‘in the eye of the observer’).	Predefined a priori quantity that drives selection and reproduction.
Selection	
Complex multifactor force based on environmental conditions, other individuals of the same species and those of other species (predators).	Viability is tested continually; reproducibility is tested at discrete times. Randomized operator with selection probabilities based on given fitness values. Survivor selection and parent selection both happen at discrete times
Genotype–phenotype mapping	
Highly complex biochemical and developmental process influenced by the environment.	Typically a simple mathematical transformation or parameterized procedure. A few systems use generative and developmental genotype–phenotype maps.

Variation	
Offspring are created from one (asexual reproduction) or two parents (sexual reproduction). Horizontal gene transfer can accumulate genes from more individuals.	Unconstrained vertical gene transfer. Offspring may be generated from any number of parents: one, two or many.
Execution	
Parallel, decentralized execution; birth and death events are not synchronized.	Typically centralized with synchronized birth and death events.
Population	
Spatial embedding implies structured populations. Population size varies according to the relative number of birth and death events. Populations can and do go extinct.	Typically unstructured and panmictic (all individuals are potential partners). Population size is usually kept constant by synchronizing time and number of birth and death events

The essence of an evolutionary approach is to equate possible solutions to individuals in a population, and to introduce a notion of fitness on the basis of solution quality. Analogous to natural evolution, an evolutionary algorithm can be thought of as working on two levels. At the higher level (the original problem context), phenotypes (candidate solutions) have their fitness measured. Selection mechanisms then use this measure to choose a pool of parents for each generation, and decide which parents and offspring go forward to the next generation. At the lower level, genotypes are objects that represent phenotypes in a form that can be manipulated to produce variations. Genotype–phenotype mapping bridges the two levels. At the genotypic level, variation operators generate new individuals (offspring) from selected parents. Mutation operators are based on one parent (asexual reproduction) and randomly change some values. Recombination operators create offspring by combining values from the genotypes of two (or more) parents. Evolutionary algorithms are easily transferable from one application to another because only two components are problem dependent: the way that the genotypes are converted to phenotypes and the fitness function. The history of evolutionary computation has shown that suitable combinations of a few simple data structures can represent possible solutions to a huge variety of different problems. In other words, a relatively small collection of possible genotypes can accommodate many different kinds of phenotypes. Just as the genetic mechanisms underpinning natural evolution are largely species independent, acting on DNA or RNA, so too in evolutionary computation the choice of suitable variation operators depends solely on the data structure present in the genotypes and not on the specific problem being tackled. However it should be noted that just because an algorithm is formally suitable, it does not necessarily mean it will be successful. Suitability only means that the evolutionary algorithm is capable of searching through the space of possible solutions of a problem, but gives no guarantees that this search will be either effective or efficient [68].

It could be argued that evolutionary algorithms are not faithful models of natural evolution. However, they certainly are a form of evolution. As Dennett said "If you have variation, heredity, and selection, then you must get evolution" [69]. This is one of the most important motivations for this research work.

3.2.3. *Problem Solving Approaches and Evolutionary Theory*

Solving a problem is the transformation of a given situation into a desired situation or goal. *Problem* can be defined by tasks specified by a set of actions, a goal, an initial state and a set of reachable states. For such problems, a *solution* is composed of a sequence of actions necessary to achieve this goal. In solving such types of problems it will be fair to co-relate natural evolutionary processes where organisms are 'solutions' to 'problems' set up by the environment. Also, as it has been studied and observed that nature has solved many problems leading towards the generations of robust organisms. In biological evolution, each mutation reconfigures the gene. Similarly, in problem solving, each interaction reconfigures the problem state. This reconfiguration may increase or decrease the difference between the reconfigured and goal states and, thus, the distance (number of ticks or steps) required to reach the goal state. Each interaction, then, has a positive (acceleratory) or negative (deceleratory) impact on the problem-solving process. In other words, each interaction aims to perform a telic function, i.e., it operates to reduce the difference between the current, problematic state and a specified goal state. Thus, one can view problem-solving interactions as operators, goal directed actions, performing a means-ends analysis in the problem space [70].The process of natural selection is a feedback process that '*chooses'* among *'alternative designs*' on the basis of deciding how good the respective modulation is. As a result of this natural selection we find *adaption*. This is a process that *constantly tests the variations among individuals in relation to the environment* -if adaptions are useful they get passed on; if not they'll just be an unimportant variation. Based on these principle it's an interesting research work to design of efficient probabilistic problem solving strategies [71]. The technique is referred to as 'evolutionary problem solving' because its strategy is inspired from principles of biological evolution. The situation description, common to dynamic programming [72] and reinforcement learning [73], is perhaps in this case a better model of natural processes. These are discrete-time dynamic systems whose state transition (environment) depends on a control (organism), which again depends on the current state.

In general understanding, to solve a problem there must pre-exist a description of the situation, operators for changing the situation, and an evaluation to determine whether the goal has been achieved. These types of problem solving strategies have been identified in engineering design with definition of steps to be followed similar to evolutionary strategies (refer [74], [75] and [71] for more details). In particular, the steps may be interpreted as shown in Table 3-4.

Table 3-4 A stepwise approach to evolutionary problem solving

problem-solution abstraction

1. *identifying the problem;* understanding and finding the problem
2. *representation;* design a data structure for the problem-solution

evolutionary search

3. *variation (initialization);* devise a plan or modify an existing one
4. *genotype to phenotype mapping;* carry out the plan
5. *selection (fitness evaluation);* evaluating its utility
6. *self-adaptation;* learning from the experience of solving

However, an important point to be made here is that solving problems is not about ***random search in the problem-solution space***, but is a search space in which the actors bring in knowledge which when sufficient the search is unnecessary. Hence, the types of problem solver under consideration are the ones that will not limit itself to direct search but also to knowledge search. This is very similar to the way living organisms act in the world i.e. a living system must also necessarily know something in so far as it can act in the world.

The hypothesis that embedding the principles of evolution within computer algorithms can create powerful mechanisms for solving difficult and/or poorly understood problems is now supported by a huge body of evidence. Evolutionary problem solvers have proven capable of delivering high-quality solutions to difficult problems in a variety of scientific and technical domains, offering several advantages over conventional optimization and design methods (refer [76], [77] and [78] for more detailed discussions). One appealing example from the design domain concerns X-band antennas for the NASA Space Technology 5 (ST5) spacecraft [79]. The normal approach to this task is very time and labour intensive, relying heavily on expert knowledge. The evolutionary-algorithm-based approach not only discovered effective antenna designs, but could also adjust designs quickly when requirements changed. Evolutionary algorithms have also been successful in many other aeronautical and aero- space engineering endeavours. Problems in this field typically have highly complex search spaces and multiple conflicting objectives. Population-based methods such as evolutionary algorithms have proven effective at meeting the challenges of this combination. In particular, so-called multi-objective evolutionary algorithms change the selection function to explicitly reward diversity, so that they discover and maintain high-quality solutions representing different trade-offs

between objectives - technically, they approximate diverse segments of the Pareto front [80]. Another case is mining the ChEMBL database[1] (which contains bioactive molecules with drug-like properties), a set of transformations of chemical structures was identified that were then used as the mutation operator in an automated drug-design application [81]. The results showed clear benefits, particularly in accommodating multiple target profiles such as desired polypharmacology. This nicely illustrates how other approaches, or existing knowledge, can be easily co-opted or accommodated within an evolutionary computing framework.

Specifically in the case of CPS, the adaptation is macroscopic behaviour shared across physical and cyber world. In other words, a CPS should be able to change its behaviour in response to environmental and internal feedback (computational cycles), often in an attempt to achieve a goal or objective. These types of goal-seeking adaptations that occur on a collective scale and/or over multiple iterations can emerge as evolution. Modern synthesis on computational theory depicts that the prevailing theory of evolution, combines Darwin's theory of variation and natural selection with Mendel's theory of genetics to characterize evolution as a process of development or change over time [82]. Another important aspect of CPS of cognition, so CPS will require having cognitive properties, which can be achieved by insertion of specific knowledge either by injecting domain knowledge or as a result of automatic learning. Domain knowledge may be optimized not only for a single problem instance, but also for the problem class. The search for domain knowledge is also subject to the *no free lunch theorems* (ref [83] for understanding this theorem) and therefore, it becomes necessary define a robust cognition architecture and also the methodology to provide knowledge about such cognitive system back into the CPSs, with necessary annotations or mappings. In the scope of this thesis work though, the ultimate interest is efficiently finding effective solutions to specific problem classes bounded by the domain of interest thus bounding the domain knowledge to some extent. This, aspect is partially addressed by the research paper [84], to make use of the generated information to identify pertinent events and to make decisions that, in turn, affect the physical space.

3.3. Evolutionary Systems

Computational systems undergo changes with the additional requirements caused by different factors. They can be viewed as evolvable at many levels of abstraction, from the rather low-level computational step evolution of a hardware system state, or of a program's computation state, to whole or partial system change or reconfiguration that may occur in the maintenance or installation of software or hardware updates. Such

[1] https://www.ebi.ac.uk/chembl/

timescales of change are studied as 'Software Evolution', see e.g. [85]. The evolutionary nature of businesses, business processes and their computational modelling is a good example. However mathematical theories of computation have largely ignored these various levels of system evolution; attention has focussed instead on developing models of computation to support effective reasoning about fixed sequential, parallel and distributed software and hardware [86]. Now, when we extend the computation layers into the physical layer towards cyber physical systems the trends for evolution becomes trivial for the system. Often, present-day systems are limited by their ability to:

(1) sense the surrounding environment in which they are operating,

(2) look ahead and anticipate events, and

(3) control system behaviour.

In an effort to relax, or even remove, these constraints, evolutionary cyber physical systems are incorporating advances in computing, sensing, and communications. The issue for our research is that when evolution is being an intrinsic feature of CPS, how can we specify and reason formally about the changes that the systems undergo, so as to define a self-evolutionary path. The introduction of evolutionary behaviour allows considerably more freedom in the way that systems may behave and it is not at all obvious that traditional methods of specification, and standard approaches to program logics and semantics, remain adequate.

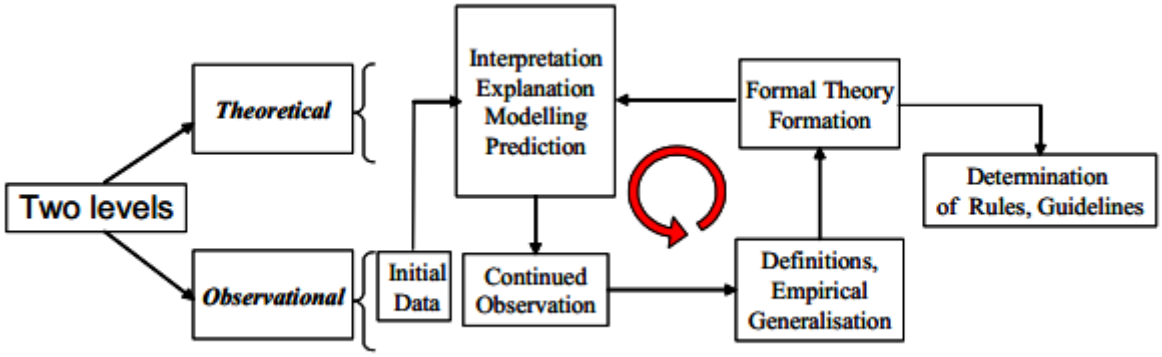


Figure 3-4 An approach to the formation of a theory of software evolution

But the evolution in systems can be systematically studied, changes can be monitored and the system can learn through the changes that the system undergo. For instance Self-adaptive systems (for further reading see[87])react to faulty situations automatically. These adaptive actions can be the learning path for system to undergo evolution so that in future there will not be the need for adaptive measures, thus making the system more resilient to failures. So, studying the evolutionary behaviour of systems is a major focus of this research work. In the context of software evolution there are, however, two aspects to such study as discussed in [88] viz. nounal and verbal. Nounal view of evolution focuses on the nature of evolution, its causes, properties, characteristics, consequences, impact, management, control and exploitation. While the verbal is concerning oneself

with providing and improving means, processes, activities, languages, methods, tools for example, whereby evolution is implemented. While, [85] provides an approach to the formation of a theory of software evolution which is as shown in Figure 3-4. This approach clearly marks two approaches i.e. Theoretical and Operational, which will also be the basic approach towards the theory for evolution for CPS.

3.3.1. *SELF-EVOLVING SYSTEMS*

A configuration of a system may be called "fit" if it is able to maintain or grow given the specific configuration of its environment [89]. An unfit configuration, on the other hand, is one that will spontaneously disintegrate under the given boundary conditions. Different configurations can be compared as to their degree of fitness, or likeliness to survive under the given conditions imposed by the environment. Thus, adaptation can be conceived as achieving a fit between system and environment. Self-Organization (SO) may provide an essential progress in the realization of embodied control. Viewing a robot in its environment as a complex dynamical system SO can help to let highly coordinated and low dimensional modes emerge in the coupled system of brain, body and environment. In this way, instead of being programmed for solving a specific task, the robot may find out by itself what its bodily affordances are, focusing only in a second step on the exploitation of the emerging motion patterns—by guiding the SO process into the directions of potential benefits [90].

Every self-organizing system adapts to its environment, in which the particular stable configuration it reaches by definition fits its particular circumstances. Let us take an example as discussed in [89], which take the pattern and speed of flow in the Bénard rolls. The speed and flow will be adapted to the specific temperature difference between bottom and surface, whereas the orientation of the spins will tend to be parallel to any outside magnetic field. In this sense, self-organization implies adaptation. This becomes even clearer if we choose a different boundary to distinguish system from environment. As noted by Ashby [91], if we consider a particular part of the original, self- organized system as the new "system", and the remainder as its "environment", then the part will be necessarily adapted to the environment. For example, in the magnet the orientation of a particular segment of spins will be adjusted to the magnetic field generated by the rest of the spins. For a given boundary, adaptation becomes less trivial when the boundary conditions change. For large changes, this in general means that the existing configuration becomes unstable. This may lead to its disintegration, and the need to start the process of self-organization anew. This may not seem a great problem for systems like the magnet or the Bénard rolls, but it would be disastrous for more complex systems such as organisms, ecosystems or societies.

Fast error detection, fault tolerant system designs and new planning strategies are required to cope with the increasing failure rates of microchips owing to continuous shrinking of devices, as well as reliance on unreliable sources of information (e.g.,

information sent by other vehicles). Some of these problems can be solved by knowledge-based techniques, such as autonomous reconfiguration and substitution of faulty subsystems and components by using system ontologies [92]. An alternative method is to equip CPS with monitors and to predict emergent behaviours at runtime. This approach makes CPS self-aware, opening up new approaches to designing systems that can dynamically reconfigure themselves in order to adapt [93] to different circumstances. However, monitoring introduces a runtime overhead that may alter the timing-related behaviour of the system under scrutiny. In applications with real-time constraints, overhead control strategies may be necessary to reduce the overhead to acceptable levels by, for example, turning on and off the monitoring. Gaps in monitoring, however, introduce uncertainty in the monitoring results. Hence, our current research [93] also focuses on efficient techniques to quantify this uncertainty and compute an estimate of the current state of the system.

3.3.2. *KNOWLEDGE EXTRACTION AND MACHINE LEARNING*

As has been discussed in previous sections, CPS has to deal with huge amount of real time data collected via wide range of sensors and are very often fed back to the system after data processing for varying purposes. As discussed in [94], a special case of CPS in manufacturing i.e. Product Line Engineering (PLE), it is clear that the next generation CPS have interstice characteristics of Runtime configuration, temporal variability, feature interaction and /Component Interaction . So, in the scope of this doctoral research work it is important to define and implement knowledge extraction and learning methodology specific to the CPS.

Knowledge extraction over the real time data that the CPS has to deal with and the actions being performed for system adaption based on the feedback loop is the challenge that eCPS will have to deal with. Knowledge Representation and Reasoning (KRR) theories of Computational Logics will be one of the strategies to be adopted because of the possibility to define well-formed semantics based framework. Artificial Neural Networks (ANN) models from AI are also an important aspect for machine learning in CPS, mainly because of the uncertain nature of the data. But at the same time considering multi-layered architecture of CPS expanded over various processing components, it is important to preserve the semantics of the data being processed at various stages. Some of the interesting works in ANN with semantics are discussed in [95–97]. Fuzzy logic is thus important aspect for this research work, which has been widely studied in works like [98,99] for data driven systems, which is a basic foundation of CPS.

Based on the discussion above, it can be highlighted that interpretability of the knowledge (usually in the form of rule-base) generated automatically from the data in real-time and use it for self-learning based on the principles of machine learning is vitally important towards eCPS. The book “Evolving intelligent systems: methodology and applications” [100] provides a very good insight into the current state of art in this domain. But, in the

context of CPS, methodology has to be designed not only to extract knowledge over the data being fed into the system, but also of the performance and control measures of the running system, so that the knowledge base of the system can be improved based on the actions. Detecting anomalies in both the cyber and physical layers can be challenging, which has been studied in a specific type of CPS in a recent doctoral work as described in [101]. So, the major research challenges that will be address in this doctoral work are:

a) designing effective and computationally efficient learning methods;

b) interpretability of the knowledge generated by such techniques;

c) problems of cooperative knowledge generation between different components/systems

3.3.3. *Software Engineering for Self-Emerging Systems*

A self-adaptive system is able to modify its behaviour according to changes in its environment, which is the basic requirement for self-evolutionary systems. As such, a self-evolutionary system must continuously monitor changes in its context and react accordingly. Presumably, the system still needs to maintain a set of high-level goals that should be maintained regardless of the environmental conditions. But non-critical goals could well be relaxed, thus allowing the system a degree of flexibility during or after adaptation. At the same time those goals will be used for finding the right paths towards evolution for advancements. In this regard, one preliminary work is presented in the paper [102], which presents a framework that allows a more or less continuous pathway from classical concepts of von Neumann machines towards Holland's classifier systems, the α-universe, functional programming languages, Lindenmayer systems or even cellular automata. This allows entirely different computing paradigms to be viewed from a common basis and be programmed by a structured assembler code.

These questions (and others) form the core considerations for building self-adaptive systems. Requirements engineering is concerned with what a system ought to do and within which constraints it must do it. Requirements engineering for self-adaptive systems, therefore, must address what adaptations are possible and what constrains how those adaptations are carried out. In particular, questions to be addressed include: what aspects of the environment are relevant for adaptation? Which requirements are allowed to vary or evolve at runtime and which must always be maintained? In short, requirements engineering for self-adaptive systems must deal with uncertainty because the expectations on the environment frequently vary over time.

Requirements engineering for self-adaptive systems appears to be a wide open research area with only a limited number of approaches yet considered. Cheng and Atlee [103] report on some previous work on specifying and verifying adaptive software, and on run-time monitoring of requirements conformance [104], [105]. They also explain how preliminary work on personalized and customized software can be applied to adaptive

systems (e.g., [105][106] [107], [108]). In addition, some research approaches have successfully used goal models as a foundation for specifying the autonomic behaviour [109] and requirements of adaptive systems [110].

One of the main challenges that self-adaptation poses is that when designing a self-adaptive system, we cannot assume that all adaptations are known in advance that is, we cannot anticipate requirements for the entire set of possible environmental conditions and their respective adaptation specifications [111]. For a generic example, if a system is to respond to cyber-attacks, one cannot possibly know all attacks in advance since malicious actors develop new attack types all the time. Specifically in the case of CPS, it is hard to close the list of anticipated errors caused by the run-time environment of the system. As a result, requirements for self-adaptive systems involve degrees of uncertainty or may necessarily be specified as incomplete. The requirements specification therefore should cope with: the incomplete information about the environment and the resulting incomplete information about the respective behaviour that the system should be able to respond.

In order to gain assurance about adaptive behaviour, it is important to monitor adherence and traceability to the requirements during runtime. Furthermore, it is also necessary to acknowledge and support the evolution of requirements at runtime. Given the increasing complexity of applications requiring runtime adaptation, the software artefacts with which the developers manipulate and analyse must be more abstract than source code.

Some important research challenges that can be pointed are:

- *How can graphical models, formal specifications, policies, etc. be used as the basis for the evolutionary process of adaptive systems versus source code, the traditional artefact that is manipulated once a system has been deployed?*
- *How can we maintain traceability among relevant artefacts, including the code?*
- *How can we maintain assurance constraints during and after adaptation?*
- *How much should a system be allowed to adapt and still maintain traceability to the original system? Clearly, the ability to dynamically adapt systems at runtime is an exciting and powerful capability.*

Many types of adaptation techniques have been developed: architecture-based adaptation [111] that is mainly concerned with structural changes at the level of software components, parametric based adaptation that leverages policy files or input parameters to configure the software components, aspect-oriented based adaptation that changes the source code of a running system via dynamic source-code weaving, and so on. Researchers and practitioners have typically leveraged a single tactic to realize adaptation based on the characteristics of the target application. However, given the unique benefits of each approach, we believe a fruitful avenue of future research is a more comprehensive approach that leverages several adaptation tactics simultaneously.

There is a wide spectrum of the degree of automation supported by the current state of the art approaches [112]. However, in general, there are significant hurdles facing the construction of fully automatic adaptive systems, many of which are reminiscent of the challenges that the AI community has faced over the past few decades. It is imperative for the software engineering community to develop better models that incorporate the AI techniques in solving the practical problems of automatic adaptive systems.

A major challenge is to accommodate a systematic engineering approach that integrates both control-loop approaches with decentralized agent-inspired approaches. Most state of the art techniques leverage a utility function to map the trade-offs among several conflicting goals of adaptability to a scalar value, which is then used for making decisions about adaptation. However, in practice, defining such a utility function is a challenging task. Practical techniques for specifying and generating utility functions, potentially based on the user's requirements, are needed. One promising direction is to use preferences that compare situations under ceteris paribus conditions. Dynamic/run-time decision requires efficient mechanisms for gathering information about the running system and its environment. Principled approaches for efficient gathering of information at run-time are needed. An important research effort will be to understand how we can collect the necessary information for different domains and derive reusable engineering approaches.

Another important aspect for evolution is that requirements themselves to be dynamically observed, i.e., during execution. Reflection [113], [114] enables a system to observe its own structure and behaviour. A relevant research work is the ReqMon tools [115] which provides a requirements monitoring framework, focusing on temporal properties to be maintained. Leveraging and extending beyond these complementary approaches, “requirements reflection” would enable systems to be aware of their own requirements at run-time. This capability would require an appropriate model of the requirements to be available online.

Such an idea brings with it a host of interesting research questions, such as: *Could a system dynamically observe its requirements? In other words, can we make requirements run-time objects? Future work is needed to develop technologies to provide such infrastructure support?*

Based on the discussion and background research we have identified some important aspects that will lead towards engineering of flexible and dynamic systems. Component based design, high level design specifications, analysis based on semantic of system design and dynamic runtime environment are necessary to achieve systems that can go through self-evolution. The following paragraphs provide a brief discussion of these concepts.

Component model - A component model as discussed in [116] can be the basic software unit of development and deployment. It constitutes state (referred to as knowledge) and behaviour, materialized into processes. Each process executes cyclically similarly to a real-time task. To achieve component interaction, components are dynamically assembled

into collaboration groups called ensembles (e.g. a fire-fighter and all temperature sensors on the same floor, all fire-fighters active at a mission site). Within an ensemble, the components interact in terms of implicit knowledge exchange, which is handled by the execution environment.

High-level design - Invariant Refinement Method (IRM) [117], IRM is a requirements-oriented design method that facilitates modelling of ensemble-based systems. The main idea of IRM is to capture the high-level goals and requirements in terms of invariants (graphically represented as rounded rectangles), which describe the desired state of the system-to-be at every time instance. Invariants are to be maintained by the coordination of the different system components (graphically represented as rectangles). This method has been followed by is a design method followed by Dependable Ensembles of Emerging Components (DEECo[2]) based systems [116]. In the process of high-level design activity, top-level invariants are iteratively decomposed into more fine-grained sub-invariants, essentially yielding a detailed contractual design of system implementation – either in terms of local component behaviour, corresponding to a component process, or in terms of component interaction, corresponding to an ensemble.

Analysis – Analysis based on the semantics of the system during design time allows fault detection and system improvisation during design time. In this regard, DEECo is well accepted design framework that allows timing properties to be quantified via static analysis and simulations. These include end-to-end response time and estimating level of inaccuracy of perceived knowledge depending on the communication latency and physical model of sensed values (given as differential equations).

Runtime Environment – As already stated that evolutionary systems needs to have runtime environment that supports dynamic linking and configurations i.e. integration of the code changes at runtime. This has been experimented in the development of the framework jDEECo[3], an implementation of DEECo in Java. This runtime environment includes scheduling of component processes, dynamic grouping of components into ensembles, and distributed knowledge exchange. Technically, jDEECo employs gossip-style network communication to uniformly address IP-based, as well as peer-to-peer broadcast-style Wireless Personal Area Network (WPAN) networks.

[2] http://d3s.mff.cuni.cz/software/deeco/

[3] https://github.com/d3scomp/JDEECo

3.3.4. *REQUIREMENTS FOR SELF –EVOLUTION*

CPS deal with physical environment which tends to be time and context varying with the dynamics of the environment thus incorporating evolutionary behavior. Self-Evolving systems can be achieved by following the principles discussed in [118] and [119] viz.

- No hard constraints imposition on the system thus ensuing fully independent process selection and execution based on the real-time requirements.
- Need to have an inherent capability to address the new or changing set of requirements, by using the knowledge acquired over time.

For the realization of self-evolving systems, design and implementation decisions and objectives are to be set at architectural or higher level using exploited protocols, standards or specifications. At the same time the integration of knowledge representation system to capture the domain of run time environment and rules that address the operation of the overall process and their ontological relations are important to achieve evolutionary CPS (eCPS). Blueprints or templates based system requirements, design and implementation provides the necessary flexibility that the eCPS requires. Based on these discussions, we can outline the following characteristics for eCPS:

- The systems need to be fully "reconfigurable" based on platforms that can exhibit an emergent behavior.
- The reconfigurable system has to be designed by the composition of process oriented components (with high level of abstraction) so as to achieve higher level of granularity over emergent behavior.
- The functional set of the components should be represented by using formal language, which is machine readable, so that the system can automatically determine the functionality of the components, map them with the domain knowledge and use/discard the component as the system evolves.
- The change in paradigm is not addressed on the code level, but at higher level of abstraction, so that the run-time behavior of the system is not hard coded but defined by establishing relationship between components, their interactions and interfaces.

To achieve these characteristics, a number of principles, proposed in the EUPASS [120] and IDEAS projects [121] can be adopted for the paradigm of eCPS. The major objectives of eCPS are thus:

- **Optimized orchestration:** The interaction between the components comprising the overall system needs to be defined and implemented in an agile fashion. This can be achieved by adopting the principles from multi-agent based, distributed control approach with embedded controllers. So that they can act as independently and interact with other components without any inferences.

- **Adaptability:** Modularity between components allows for stepwise upgradeability and flexibility to changes. The actual system may also adapt to minor changes via its control system, which, being skill-based, allows for emergent behavior to be exploited in the global scenario.
- **Robustness:** Each of the components of the overall system is dedicated, small, independent and have their own control systems. The control system is goal-oriented, and the system is process-oriented. This results in a dedicated system based on an adaptable concept with advanced interfaces.
- **Plugability:** The components of the overall system have to be pluggable into the main work-flow of the eCPS at run time. This can be achieved by designing element which are very task-specific in order to accomplish only a simple action with well-defined interfaces for accessing the providing functionalities. The overall system is achieved by complex tasks as the union of several simple tasks.
- **Learnability:** The overall system should be able to learn from the changing environment, system dynamics and system behavior towards adaptation. The system should be able to incrementally improve the knowledge base of the overall system and perform automated reasoning to extract new facts and possible behaviors.

Fundamentally, the discussion in this section suggests that eCPS can only be achieved if the lowest building blocks of a system are those that exhibit the highest rate of adaptability and/or evolvability. [122] provides an initial draft for generic architecture for self-evolvable CPS.

4. CYBER PHYSICAL SYSTEMS

People think computers will keep them from making mistakes. They're wrong. With computers you make mistakes faster.

-Adam Osborne

In this chapter is presented one of the most important domain of research of this thesis i.e. cyber physical systems (CPS). CPS may well become the theory backing up a new wave of computing, which will be based on systems able to deliver new levels of performance and efficiency. These types of systems actively engage with the real world in real time and thus require a new understanding of computing as a physical act together with computing activities i.e. understanding of computers beyond information and cyberspace. This, chapter thus builds the foundation for understanding the CPS domain by analysing different aspects of CPS design, modelling and implementation. This chapter provides readings on enablers of CPS and relation with other scientific domains. The chapter also makes study on model based engineering and software engineering principles that are deemed important for CPS. The chapter ends by analysis of programming landscape for CPS with discussion different programming paradigms and providing brief discussion on class of programming languages that have been selected for the implementation of the research work.

CPS research is still in its infancy, in the sense that research has been portioned into isolated sub-disciplines, which form the back bone for CPS. Various researches activities in domains such as sensors, communications and networking, control theory, mathematics, software engineering, and computer science is the foundation for CPS but streamline research towards unified domain of CPS has just begun. For example, systems are designed and analysed using a variety of modelling formalisms and tools. Each representation highlights certain features and disregards others to make analysis tractable. Typically, a particular formalism represents either the cyber or the physical process well, but not both. Although this approach to modelling and formalisms may suffice to support a component-based "divide and conquer" approach to CPS development, it poses a serious problem for verifying the overall correctness and safety of designs at the system level and component-to-component physical and behavioural interactions [123]. In the following sections we will explore various aspects of CPS and will discuss and identify the research needs for CPS.

The term CPS refers to a new generation of systems with integrated computational and physical capabilities that can interact with humans through many new modalities [25], [124]. The ability to interact with, and expand the capabilities of, the physical world through computation, communication, and control is a key enabler for future technology developments. This paradigm opens up opportunities and research challenges towards smarter and critical design and development of next-generation airplanes and space vehicles, hybrid gas-electric vehicles, fully autonomous urban driving, prostheses that allow brain signals to control physical objects etc..

For wider understanding of CPS, Figure 4-1 shows the concept map as conceived by S. Shyam Sunder of NIST at the NIST CPS Workshop on March 13, 2012 in Chicago. This

concept map clearly indicates that CPS are a form of control systems that extends the existing control theory to embrace the dynamics of software and networks, which can have profound effects on stability and dynamics of the physical subsystems and vice versa. In CPS physical component will interact with the cyber component via sensors and actuators thus requiring the need for applications and algorithms to deal with real-time data processing, adaptive architectures and intelligent decision making. The control strategies implemented in the cyber subsystems need to be adaptive (responding to changing conditions) and predictive (anticipating changes in the physical processes). The concept map also depicts the wider range of application of CPS, which are also highlighted in different research papers like [125], [126], [127], [11] and [128].

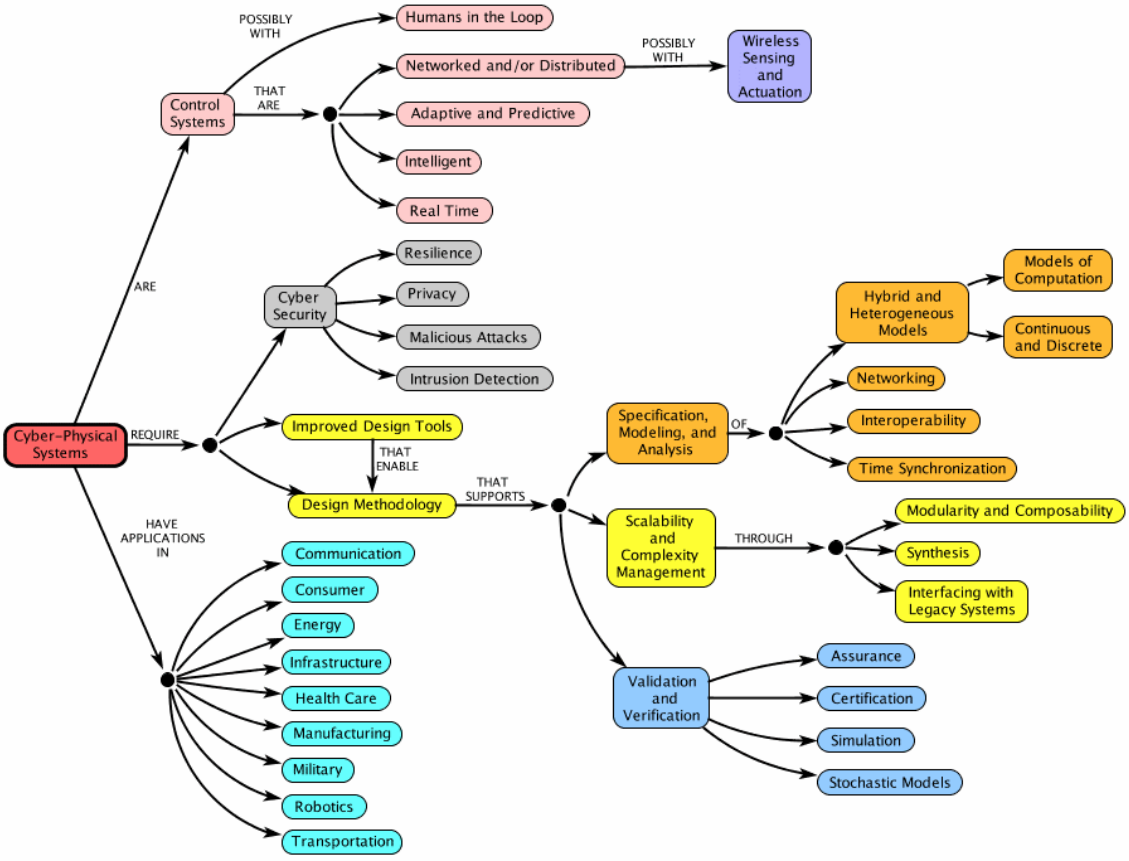


Figure 4-1 Cyber Physical System Concept Map[4]

[4] http://cyberphysicalsystems.org/ This chart began with a taxonomy given by S. Shyam Sunder of NIST at the NIST CPS Workshop on March 13, 2012 in Chicago. Edward A. Lee of UC Berkeley converted the taxonomy to a picture and then started evolving it with input from various people. So far, direct contributions to the current version have been made by: Philip Asare, University of Virginia, David Broman, UC Berkeley, Edward A. Lee, UC Berkeley, Martin Torngren, KTH, S. Shyam Sunder, NIST

From the concept map the major focus of our interest is in the "***REQUIRE***" node that identifies the various domain of research for CPS. Different sub domains of this node will be discussed in details in the following sections, but in summery CPS will have the need for tools and methodologies for requirements specifications, simulation, analysis and modelling systems by taking into consideration of both the physical and computational world.

4.1. Enablers of Cyber Physical Systems

Over the years, systems and control researchers have pioneered the development of powerful system science and engineering methods and tools, such as time and frequency domain methods, state space analysis, system identification, filtering, prediction, optimization, robust control, and stochastic control [25]. At the same time, computer science researchers have made major breakthroughs in new programming languages, real-time computing techniques, visualization methods, compiler designs, embedded systems architectures and systems software, and innovative approaches to ensure computer system reliability, cyber security, and fault tolerance. Computer science researchers have also developed a variety of powerful modelling formalisms and verification tools. CPS research aims to integrate knowledge and engineering principles across the computational and engineering disciplines (networking, control, software, human interaction, learning theory, as well as electrical, mechanical, chemical, biomedical, material science, and other engineering disciplines) to develop new CPS science and supporting technology.

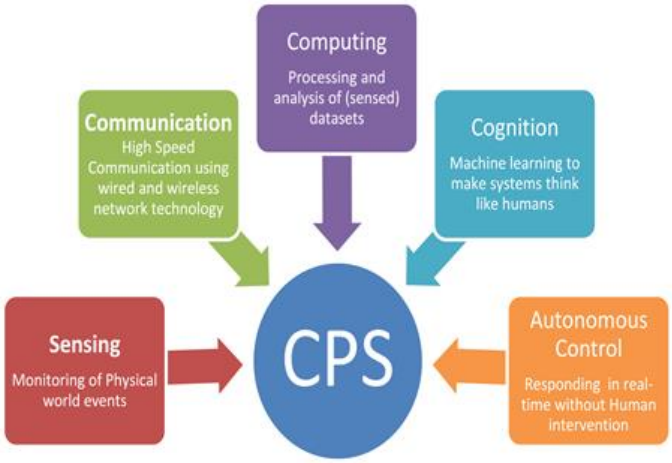


Figure 4-2 Important enablers of CPS

Technological and theoretical underpinning for CPS requires research in different aspects such as Sensing, Communication, Computing, Cognition and Autonomous control, which are as shown in Figure 4-2. Sensing and communication are one of the most important characteristics of CPS which builds the interface with the real-world environment. In the context of CPS its necessary to address cross-domain sensor sources and data flows i.e. multiple types of sensors will be integrated in CPSs. Moreover, these cross-domain

sensing data will be exchanged over heterogeneous networks. As also identified by NIST CPS workshops, the CPSs should be able to collect, analyse, process, and react to the many types of sensing, communications, and other data types that will be captured during service operations [6]. But, the design and implementation of networked control systems pose several challenges related to event-driven computing, software, variable time delays, failures, reconfiguration, and distributed decision support systems [25]. Another important aspect is cognition

On the other hand, CPSs are part of the information systems that has to deal with ever increasing amounts of available information, both valuable and useless and react over it. In such situations, it's not always possible to keep human in the loop to make decisions to react to the detected situation. In addition, it is expected that, since they offer functionalities that cannot be realized with conventional technology, CPSs will increasingly be deployed in safety-critical situations (some of such situations are described in [129], [130], [131] and [132]). So, CPS must be able to detect infrequent scenarios (e.g. with a probability of perhaps once in 500 years per driver/pilot/physician) in order to assess the likelihood of accidents or fatalities. This requires to have comprehensive cognitive behaviours under varying situations including emergency or stressful scenarios [133]. Some form of intelligence in embedded system such as adaptive sensing, clock synchronization and intelligent localization of distributed embedded units are some basic problems that can be suitably addressed by considering automatic and conscious intelligent mechanisms [134].

Autonomic Computing is another important aspect and the vision of Autonomic [135] refers to the tendency of computers to become ubiquitous. In the scenario of complex systems like the case of CPS, it tends to form large network of computing devices and smart objects with complex and multiple interactions, leading to increasing difficulty to manage. Thus, autonomic computing in general inherits the necessity of systems that will be able to take of it with minimized user interactions and need for reprogramming. Even though each component of CPS (or autonomous systems) are modules of fine granularity with self-computational power, it is necessary to find new ways of coordination and automatic plugability [136].

It is also very important to state that during the definition and specification of CPS it is very important to consider that CPS are intrinsically concurrent i.e. the cyber and the physical subsystems coexist in time. A model of a CPS comprises models of physical processes as well as models of the software, computation platforms and networks. The feedback loop between physical processes and computations encompasses sensors, actuators, physical dynamics, computation, software scheduling, and networks with contention and communication delays. Modelling such systems with reasonable fidelity is challenging, requiring inclusion of control engineering, software engineering, sensor networks, etc. [137].

4.1.1. *RELATED SCIENCE BASE*

The behaviour of CPS is characterized by the nonlinear interaction between discrete (computing device) and continuous phenomena (the physical substratum). For this reason, research on hybrid systems plays a key role in modelling and analysing CPS. CPS are usually spatially distributed and they exhibit emergent behaviours (i.e. traffic jams, cyber-attacks), which result from interactions among system components, and which cease to exist when specific components are removed from the systems. Owing to their ubiquity and impact on every aspect of our life, one of the greatest challenges of this century is to efficiently predict the emergent behaviours of these systems. The complexity of their models, however, often hinders any attempt to exhaustively verify their safe behaviour.

CPSs require a transdisciplinary approach or interdisciplinary, multidisciplinary and transdisciplinary at the same time [127]. CPS involves two knowledge domains (i.e. the cyber and physical domains), multidisciplinary involves more than two knowledge domains (for CPSs, additional domains such as biological, engineering and information sciences can be considered), and transdisciplinary extends the knowledge from the various domains towards implementation and application, for instance by providing architectures and technologies to realize the artefacts and services within the CPS. This CPS-specific interpretation of transdisciplinary seems to be in agreement with Pohl's description of transdisciplinary research, which he says 'is not only about producing knowledge but it is also problem- and solution- oriented, and the research results are translated into usable products' [138].

In general To achieve robust CPS and also eCPS, the modular design and implementation methodologies are very important i.e. each of the smallest possible components have well defined capabilities and interaction with other components. The main issue to be addressed in this section is describing the areas in which CPS control systems are getting inspiration to solve the requirements at fine granularity. Numerous scientific domains investigating phenomena which CPS also can provide helpful tools and valuable background to cope with the complexity of manufacturing systems [139], [140], [141]. Some important research domains from which the principles and results can be adopted for the realization of eCPS are:

- **Complexity Theory**

  Complexity Theory looks for simple causes leading to complex behaviors [118]. All CPS inherit the characteristics of complex systems i.e. they are spatially and/or temporally extended non-linear systems with many strongly-coupled degrees of freedom. eCPS is composed of numerous simple modules which are connected to each other and have multi-lateral interactions. The independent components have high degree of freedom, yet in the run-time environment they have to function within the global constraints to achieve the system functionalities.

- **Natural life and Artificial Life**

  Evolution has been long studied in the natural life by many prominent biologists. The principles and theories from natural life are useful to create life-like behaviors with the capability of evolution on computers and other "artificial" media. eCPS are very similar to artificial living systems which have modifiable structure and exhibit some kind of self-organization, adapt to their environment and react to stimuli. They are capable of evolving according to the circumstances, namely in terms of state of available resources and formulate new configuration to address the changes in the overall paradigm. Emergent behaviour of an ecosystem of complex computational systems is studied in [142] and provides an interesting discussion on works that have been inspired from natural sciences.

- **Multi-Agent Systems**

  In the paradigm of CPS, the presence of wider aspect of contextual information, the units interacting with the system need to be modeled as agents which can be a human, an association, an animal, or a piece of software, sensors or network of sensors, tagged objects etc. which are eventually connected to some computational components. The fundamental characteristics of such agents are identity, intelligence and the ability to act and react in order to persecute goals. Agents have at least a certain degree of autonomy and can compete or collaborate with others, which will provide the foundation for reconfigurable orchestration to meet the needs in the change in context. [143] provides an interesting interlink between evolutionary systems and multi-agent systems.

- **Semiotics**

  Semiotics concerns the study of signs/symbols in three basic dimensions: syntactic (rule-based operations between signs within the sign system), semantics (relationship between signs and the world external to the sign system), and pragmatics (evaluation of the sign system regarding the goals of their users) [144]. This principle, even though originates from the biological science provides strong theoretical foundation that can be adopted in the area of CPS. [89] provides and interesting discussion on interlink between semiotics and AI, which will be an interesting foundation for CPS.

- **AI and Reasoning**

  AI has been a foundation for the studies for many machine learning methodologies and algorithms while various reasoning techniques have been developed for formal knowledge representation methodologies. Since, learning is the important preceding step of evolution theoretical foundation on AI and Reasoning is very important aspect for CPS. KEEL (Knowledge Extraction based on Evolutionary Learning) tool, an open source software that supports data management and provides a platform for the analysis of evolutionary learning for Data Mining problems of different kinds including as regression, classification, unsupervised

learning [145]. Theoretical foundation on case-based reasoning will be an aspect which has been applied for adaptation (some works are described in [146,147]). And since adaptation is the first step that follows the learning, theoretical foundation on reasoning adaptation is important aspect towards CPS.

4.1.2. *CPS AS COMPLEX SYSTEM*

Network models are used to capture the essence of interaction among the many components in a system, whereas chaos theory generally attempts to model some resultant outcome, such as prices or investment. It has relevance to brains - neural networks; to organizations - networks of departments and people; and to industries - networks of enterprises [141]. Network theory also concerns the study of graphs as a representation of either symmetric relations or, more generally, of asymmetric relations between discrete objects. In this sense, classical networks are treated has graphs that may contain both active and passive nodes connected through edges (directed or not) [148].

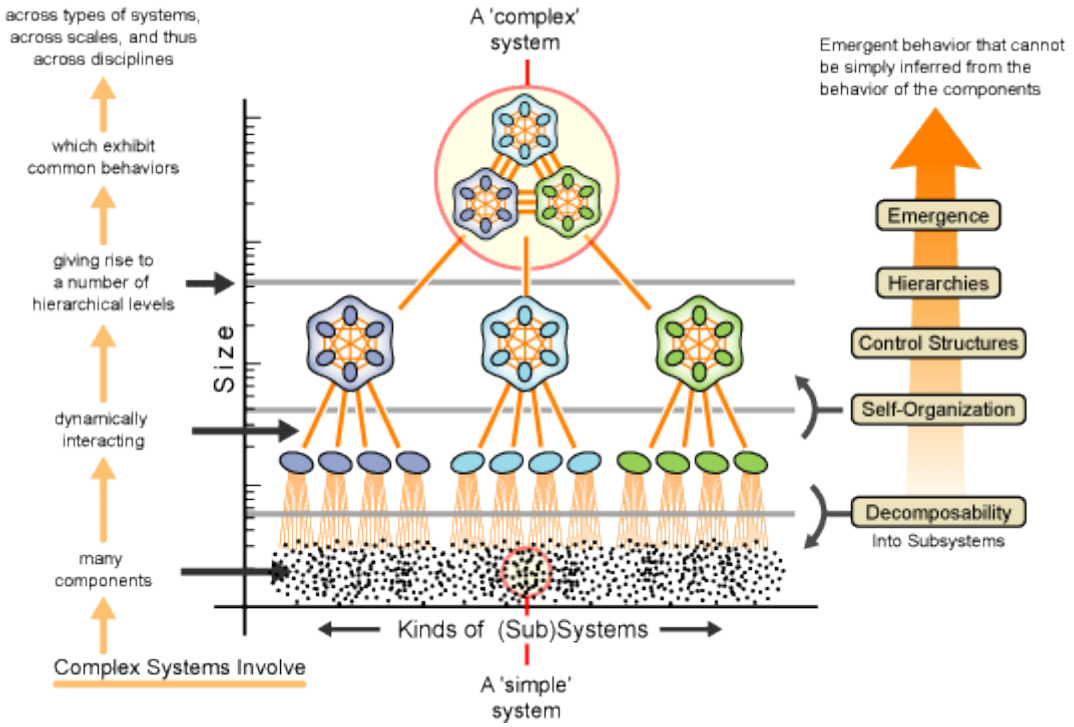


Figure 4-3 Characteristics of Complex Systems and Networks

Some of the important aspects of the recent developments in the science of network theory are:

(1) Focused on the properties of real-world networks, it is concerned with empirical as well as theoretical questions;

(2) It frequently takes the view that networks are not static, but evolve in time according to various dynamical rules; and

(3) It aims, ultimately at least, to understand networks not just as topological objects, but also as the framework upon which distributed dynamic systems are built.

The second point is the most important one for our current domain or research because CPS can be designed as complex network systems with various nodes in both computational and physical pane connected via interactions as arcs. Figure 4-3 which is referenced from [149] shows the characteristics of a complex system in general. The most important characteristics of importance for us are emergence i.e. the emergent behaviour that is not only inferred from the behaviour of components but also by the execution environment of CPS as complex system.

Based on the depiction of CPS as network structure, the CPS structure can itself encode a lot of information which can be interpreted for inferring useful knowledge. At the same time the same structural model can play a crucial role in dissemination of information to facilitate the exchange of information and distributed computations. [150] provides interesting discussion on mathematical under-pinnings of the diffusion of information over networks which provides an important background for our research towards self-evolutionary CPS. In building CPS, it is also important to consider towards building a resilient and integrated cyber–physical system (CPS). In the scope of this research work resilience is addressed across stability such the CPS can achieve a stable sensing-actuation close-loop control even though the inputs (sensing data) have noise or attacks, which is also discussed in the paper [151].

4.2. Modelling of Cyber Physical Systems

CPS is system of systems, where complex and heterogeneous systems interact in a continuous manner and proper regulation of it necessitates careful co-design of the overall architecture of CPSs. To address this need, it requires CPS specific modelling and programming tools and methodologies. In recent years, a number of modelling approaches and tools for CPSs have emerged which address the natural heterogeneity in CPS. Among the class of modelling tools are the meta-modelling techniques and meta-programmable tools, different formal semantics approaches like denotational, axiomatic, operational or a hybrid of these, multi-agent semantic models, event based semantic models, actor oriented design approach [152]. At the same time CPS involve physical objects and devices, CPS require a lot of multi-physics modelling, ranging across mechanics, aerodynamics, hydraulics, thermodynamics, electrical, and combinations thereof. Since CPS cannot go without control, a number of functions must be specified for CPS control, monitoring, and supervision [153], [154]. Challenges in modelling of cyber physical systems (CPSs) arise from the intrinsic heterogeneity, concurrency, and sensitivity to timing of such systems. The research paper [155], uses a portion of an aircraft vehicle management system (VMS), specifically the fuel management subsystem,

to illustrate the challenges, and then discusses technologies that at least partially address the challenges. Commonly employed design techniques are sophisticated and include mathematical modelling of physical systems, formal models of computation, simulation of heterogeneous systems, software synthesis, verification, validation, and testing. But, it's still an ongoing research to find a set of sequential steps that, if followed carefully and correctly, encompasses each of these design techniques and sufficiently governs the development of a complex cyber physical system [156].

4.2.1. *ABSTRACT MODELS*

In computer science we will find lots of reference to various entities that are characterized as abstract and various activities characterized by abstraction. For a very common example we can consider the abstract data types in programming. Programming languages facilitate varying levels of data and procedures abstraction. The author of the book "One computer science textbook even characterizes its subject matter as the science of concrete abstractions" even characterizes its subject matter as the science of concrete abstractions [157]. Abstraction is usually intended to facilitate the acquisition of knowledge about an underlying concrete reality, which are often concerned with complex systems. Abstract model theory provides an approach that allows us to study a wide range of logics, entities and their relationships, which has been well discussed in [158]. In the scope of software engineering and system design, every software system, whether it is an operating system, a networked system distributed over multiple machines, or a single user program, is a carefully orchestrated interaction of entities variously called tasks, threads, or processes. Programming languages, operating systems, and networks all exhibit information hiding or abstraction. For instance in Object Oriented Programming (OOP) an object's class is an abstraction specified by a programmer in program code.

The class of systems under study in this dissertation are CPSs which are complex systems which require lots of efforts and resources for construction. In such scenarios, abstract models of CPS can play an important role for characterizing and studying various behaviours of CPS. In this regard, the concepts of abstract machines will be very useful for the design and implementation of CPS. A typical abstract machine consists of a definition in terms of input, output, and the set of allowable operations used to turn the former into the latter. The best-known example is the Turing machine. Abstract machines are machines because they permit step-by-step execution of programs; they are abstract because they omit the many details of real (hardware) machines. The instructions of an abstract machine are tailored to the particular operations required to implement operations of a specific source language or class of source languages. An Abstract Machine is described using the Abstract Machine Notation (AMN) – which is uniform notation for providing description for different levels of system development, from specification, through design, to implementation (follow for detail understanding of AMN [159]). Abstract machines encapsulate:

- State - consisting of a set of variables constrained by an invariant
- Operations - operations may change the state, while maintaining the invariant, and may return a sequence of results.

Thus, abstract machine comprises a state together with operations on that state. In a specification and a design of an abstract machine the state is modelled using notions like sets, relations, functions, sequences etc. The operations are modelled using Pre- and Post-conditions using AMN. This methodology for defining abstract model of CPS and eCPS is followed in the scope of this thesis work to formulate a theoretical foundation of eCPS to characterize different functional properties of eCPS.

4.2.2. *Model based Engineering*

Models provide useful abstractions of the physical reality with formal properties like determinism. As, stated before, the challenges in CPS modelling arise due to the heterogeneous nature, concurrency of different physical processes, and time sensitiveness. Thus, CPS requires different levels and types of abstractions for physical and computational components and their interactions. These abstractions differ due to the underlying physics, granularity and details required. Several models of computation (MoC) for modelling interactions among actors in the CPSs exist such as continuous time (CT) [160], finite state machines (FSMs) [161], discrete events (DE) [162], process networks (PNs) [163] etc. MoCs give semantics to concurrency in the model and also define the communication semantics. This approach helps in answering several crucial modeling issues, such as whether components in the model execute simultaneously, whether they share a notion of time, whether data are exchanged using publish-and-subscribe protocols, synchronous or asynchronous message transmission etc. Also note that the heterogeneous nature of most CPS applications necessitates use of heterogeneous mixtures of MoCs. Hybrid systems approach involves integration of continuous physical dynamics expressed with ordinary differential equations with the discrete behaviours expressed using finite automata.

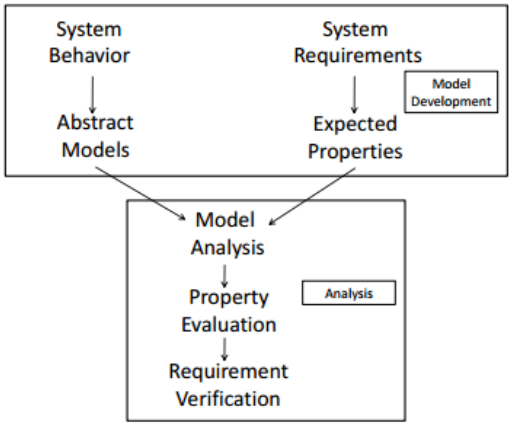


Figure 4-4 Model based engineering methodology

An efficient way to perform such design and analysis of a system is the Model Based Engineering (MBE) approach. In this approach, the designers build models or abstractions of systems' behavior and perform analytical evaluations on the model to verify different design decisions (refer [164] for detailed understanding of the approach). In general, MBE is the method of developing behavioural models of real systems and analysing the models for requirement verification. Figure 4-4depicts the different phases in this methodology.

There are two main phases in MBE:

- **model development**, in this phase, a set of expected properties of the system is determined from the system requirements. An abstract modelling is further performed that generally involves capturing appropriate parameters whose variations can reflect the system behaviour
- **model analysis**, in this phase mathematical analysis is performed on the abstract model to evaluate the expected properties and verify the system requirements.

In order to design and model a CPS, it's required to have different types of models. Some of the most commonly used models are as discussed below:

- **Functional models:** These models often use notation specific to the domain of application and describe the functionality of the system under development. In embedded systems, generally control-theoretic description techniques like Block Diagrams/Data Flow Diagrams and extensions thereof are used.
- **Platform models:** These models describe the platform the functionality is deployed to i.e. in general describing both the HW elements including control units and buses as well as the accompanying SW stack like middleware or operating systems.
- **Environment models:** These models describe the behaviour (or the part of) the environment the system under development is embedded into. In general, physical processes like the rigid-body mechanics controlled by the system are described, in these models. At the same time it involves the definition of the systems' situational and sensing model specifically in the case of CPS.
- **Information Model:** Information model provides the representation of concepts and the relationships, constraints, rules, and operations to specify data semantics for a chosen domain of discourse. Hence they provide a common understanding of the context of the domain. Information models can be static and dynamic. Static models form the meta-model of the concept while dynamic models comprise of the concepts to capture the dynamics of the system and include data from power meters, sensors like temperature or humidity, calendar schedule of conference rooms, local weather, etc.

- **Energy Model:** Energy models are used to get insights on energy flows and identify opportunities to improve efficiency. Since, CPSs are energy critical systems, it's important to specify model to estimate energy consumption by different components including the energy consumption during their interactions

Note that in the scope of this research work the models of importance are first four models, while the energy model is not explored.

Another aspect to be considered in the CPS design in the dynamicity of the system, and hence, design of CPS require handling the complexities posed by temporal variations and designing situation-specific control actions [165]. In this regard, [166] address the problem of situation based control in cyber physical environments using situation calculus. Their approach enables effectively handling temporal events and creating appropriate response actions. Another approach is proposed in [167] that uses the technique for translating the analytical dynamics of a physical system into running simulation codes. The authors discuss the syntax of the core analytical modelling language, which provides constructs for modelling both simple and complex features of the physical system along with the steps for mapping this language to the executable code.

In model-based design and model-driven development [168], models play an essential role in the design process. They form the specifications for systems and reflect the evolution of the system design. Important aspect for us here is on how can design artefacts can be part of the overall system evolution. But at the same time the intrinsic heterogeneity and complexity of CPS stresses all existing modelling languages and frameworks. A model of a CPS comprises models of physical processes as well as models of the software, computation platforms and networks. Additionally, CPS models typically involve a large number of heterogeneous components, which makes necessity for well-defined composition semantics. [169] proposes a methodology to aid engineers in the design and control of complex systems which is based on the ***system with self-organizing behaviour***. These works can important steps towards defining methodology for design and modelling of CPS that can evolve over time by actively interacting with all the artefacts used for the realization of the system.

4.2.3. *SOFTWARE ENGINEERING FOR CPS*

Software architecture is a well-accepted technique for disciplined engineering of large-scale software systems. Software architecture typically models a system as a graph of components and connectors, in which the components represent principal computational elements of a system's run-time structure and the connectors represent the pathways of communication between components [170][171][172]. These elements are annotated with properties that ***characterize their abstract behaviour and facilitate reasoning about system-level design trade-offs***. There has been considerable research and development in

architecture description languages. At the same time standardized notations such as UML 2.0 [173], SysML [174] and AADL [175] provide modelling vocabularies of components, connectors and properties. Software architectures also support reuse of design expertise and code infrastructure. In many cases an architecture of a system fits within a common family, referred to as an architectural style [176]. Software architecture has been applied effectively to numerous embedded and control systems. For instance, architecture description languages such as Meta-H and AADL have been used to model avionics systems, automotive control systems, and other applications [177]. PhD Thesis by Bauer, Kerstin [178] and Saeedloei, Neda [179] provide interesting results towards system modelling of CPS providing strong foundation for this research work.

In the context of CPS, the software has to live in an open and highly dynamic world. But in traditional software development process is based on the closed world assumption, which means that the boundary between the system and its environment is known during design-time and that the environment does not change while. In contrast, CPS in general cannot be specified completely during design time due to incomplete knowledge about, for instance, services and devices available during system operation [180]. CPSs are required to support modifications that are not envisaged at design time, as these modifications are triggered by the actual system context at run-time. These context-aware modifications require that CPSs are able to extend and modify their functionality without stopping and disturbing the rest of the system. The development of CPS therefore has inherently to live with uncertainty in the specifications [181]. During operation, such systems must frequently adapt to the executing environment changes faced at run-time and must be able to continue to behave in a controlled and safe way. This smart adaptation must be based on the system knowledge following a well-defined self-decision making process. From a software engineering point of view, dynamic composition of CPS poses a radically new situation. In contrast to ‘traditional’ embedded systems, hardwiring the system with its sensors/actuators at design-time is no longer feasible, because the concrete objects that the CPS will federate may be unknown at design-time. Also, individually and manually connecting the CPS with those objects at run-time will not scale once CPS consist of thousands or more collaborating objects [182].

Another aspect to be considered differently in the software development life cycle of CPS is requirement engineering (RE). CPSs are the class of system of systems, thus, the independence of the constituent systems and their evolutionary nature leads to exceptionally distributed requirement engineering activities for a multitude of stakeholders with isolated requirement engineering approaches. Wiesner et al. propose a RE content model for requirements elicitation and documentation at different levels as a solution [183]. Furthermore, the complexity of systems of systems leaves requirements fragmented among many disciplines and sometimes conflicting, unstable, unknowable or not fully defined. Finally, the properties of systems of systems emerge from the cumulative interactions of the single systems. Therefore, RE methods and tools have to be able to verify emergent effects against requirements with predictable results.

Verification and Validation (V&V) methodology for CPS also requires systematic evaluation of the entire software system. In contrast to, for instance, web-based systems, a CPS interacts with its physical surrounding by perceiving data thereof and derive decisions for actions therein – like driving trajectories for self-driving vehicles. The interaction with the physical world (which is of continuous nature and inherently uncertain) and the dynamicity and open-endedness of CPS makes their testing and verification a difficult challenge. Christian Berger has addressed this challenge and presented an approach for regression testing of software modules that based on resource-isolation; thus, individual test cases are wrapped into isolated process contexts allowing their parallel execution without mutual interference [184]. Zheng and Julien [185] presented interesting results of an empirical study on the demands on tools and techniques for verification and validation of CPS. The initial conclusions are:

(i) trial-and-error testing (state of the practice) does not provide sufficient rigor in error detection,

(ii) formal methods provide a desired level of expressiveness but are neither intuitive nor efficient, and

(iii) existing simulation tools are limited in their capabilities to jointly model physical and cyber components.

In this regard, Mitsch and Platzer present ModelPlex [186], which is a principle to build and verify high-assurance controllers for safety-critical computerized systems that interact physically with their environment. It guarantees that verification results about CPS models transfer to the real system by safeguarding against deviations from the verified model. But, the work in ModelPlex doesn't consider the V&V model of software development methodology thus its necessary to have this tool integrated with standard V&V model. This, fact can also be related to the statement made in [185] –"*CPS developers are generally unfamiliar with traditional software verification and validation tools and methodologies.*" . Also, the formal model approaches are insufficient to meet CPS applications' needs. There is a significant gap in language between formal models of computing and communications and models of physics that makes applying them jointly in CPS challenging. The tools and techniques available to developers have largely focused only on the computational aspects, leaving physics and physical models underrepresented in the verification and validation stages. An interesting work in the similar domain is also discussed in [187], from a SOA view point.

At the end, it is necessary to link up with agile software development methodology, which is a widely accepted methodology in the domain of software engineering (refer [188] and [189] for detailed understanding of the methodology). But, agile are still rarely applied in a multi-disciplinary context, and mainly used within the software development. Stelzmann provides an interesting study on the differences between context of agile software and hardware systems by analysing prior work about the context of agile software development and conduct interviews in system developing companies [190].

The main conclusion is that ‘in contrast to software, hardware systems that have to be produced physically often are difficult to be developed in small cyclic steps’. In addition, it is stated that ‘only if prototyping, testing, and implementing changes can be done quickly and cheaply, this principle is feasible’. This is mainly because developers of this domain often embrace a transformative approach that encompasses the entire product development cycle. However, the benefits of agile software development are real, especially for companies looking to accelerate the product development cycle, a common refrain in many industries that use embedded software. With careful adaptation of agile design and development practices, software teams can overcome the inherent constraints of embedded software development and successfully accelerate new product development. In this domain an interesting enhancement on agile methodology specifically targeting embedded software development are discussed in [191], that provides nine strategies for adopting agile methodology with enhancements on agile methods, agile team structures and behaviours.

4.3. PROGRAMMING LANDSCAPE FOR CPS

CPSs are resources constrained systems i.e. resource constraints (e.g., CPU, memory, storage, energy etc.) are a major issue in developing and debugging CPS. It has been reported by Zheng and Julien, from the survey that high level programming language, along with functional languages, are actually quite commonly used in developing CPS applications [185]. In order to understand the programming paradigm for CPS, let’s divide the programming landscape into two aspects: programming paradigms, and program organization. The following subsections provide the detailed discussion on both the aspect.

4.3.1. PROGRAMMING PARADIGMS

Programming paradigms describe the style and methodologies used to solve software engineering problems. Paradigms differ in the concepts and abstractions used to represent the elements of a program. As new technologies, platform, and applications evolve, new paradigms may be needed to maximize the efficiency of the software development process. In the scope of this work, we divide the programming paradigms landscape in two classes; resource-oriented paradigms, and interaction-oriented paradigms.

4.3.1.1. RESOURCE ORIENTED

Resource-oriented paradigms direct the focus of software developers to resources when designing the application. Principles of resource abstraction, encapsulation, and composition guide the design process of resource oriented applications. This class of programming paradigms encourages the modelling of application entities as hierarchical

families of abstract resources. We classify the resource-oriented paradigms into two subclasses according to the type of the resource.

1) **Passive Resource:** In this class of programming paradigms, the software system is composed of passive resources i.e. resources don't have an independent control or execution context. For example, data-oriented paradigm [192], and object-oriented paradigm [193]. In WSNs, a number of data-centric architecture has been proposed where the network is viewed as a distributed database management system. Applications uses a high-level 4GL language to access the network. This greatly simplifies application programming, although it lacks flexibility and control over the network resources and application performance such as TinyDB [194].

2) **Active Resource:** In this class of programming paradigms, the software system is composed of active resources i.e. resources have independent control and execution context. In recent years, integrating the Service Oriented Architecture (SOA) with sensor networks gained much attention from researchers around the world [195–197]. SOA realizes a loosely coupled software architecture that allows software resources to be plugged in and out on demand, and allows composition of complex services from simple ones. It is suitable for resource-centric environments where services represent a convenient abstraction of resources. OASIS is programming framework and middleware for service oriented sensor network [198]. The REpresentational State Transfer (REST) architectural style, introduced by Fielding[199], defines a uniform connector interface for globally identified resources. REST has been applied successful in the World Wide Web. It is characterized by interface genericity for making the system components expose the same minimal interface. This allows all interactions to be resource-generic rather than resource-specific [199].

4.3.1.2. INTERACTION ORIENTED

Interaction-oriented paradigms direct the focus of software developers to the interactions among distributed nodes to collect, organize and comprehensively utilize available resources. In this class of paradigms, the resource aggregation logic is effectively separated from the business logic. The programming constructs provide abstractions for interaction aspects like collaboration processes, team plans, roles, agents, contracts and missions. Interaction-Oriented Programming (IOP) was originally proposed to construct complex multi-agent systems. It enables expressive declarative specification of agent interactions to channel the intellectual energies of designers into the most effective design tasks [200]. The Role-Oriented Adaptive Design (ROAD) was proposed as a general-purpose paradigm for adaptive software systems [201]. ROAD provides composite applications the ability to adaptively and continuously reconfigure itself to remain viable as environment and run-time conditions undergo changes. ROAD exploits contracts to

realize adaptive service compositions that can be dynamically constructed by connecting and configuring other services.

4.3.2. *PROGRAM ORGANIZATION*

Program organization refers to the scheme used to organize different sections of executable programs for execution. We classify the program organization landscape into two classes; monolithic, and modular.

4.3.2.1. MONOLITHIC:

The term monolithic is traditionally used to describe programs that do not have any form of modularity; but within the scope of this work use the term to describe program runtime organizations that have a single program image executed by all nodes. Monolithic programs are built as a single, unstructured, self-contained software unit. Most existing wireless sensor networks are based on monolithic applications. We classify the monolithic programs into two subclasses according to the type of control.

1) Static Control In this class, the logic control is statically specified in the monolithic program image. Programs in this category are usually designed for a very specific task that is unlikely to change. It provides the least flexibility of all types of program organizations.
2) Dynamic Control In this class, the logic control is external to the monolithic program image. The program is designed to provide a number relatively generic commands, or services that can be invoked in the order and configuration specified by an external program. For example, an interpreter maybe implemented as monolithic program image, but the control flow is determined by the interpreted scripts. The same interpreter can run multiple scripts with different control flows. Furthermore monolithic program with dynamic control can be divided into two classes according to the scope of control.
    i. Host Centric: In this class, the program control is remains on a single host, such that control script is completely executed on that host. Virtual machine based platforms for WSNs belong to this class. In such platform, a bytecode interpreter supports the execution of applications. It achieves energy efficiency by providing a concise way to represent the application logic, although it suffers from instruction interpretation overhead.
    ii. Network Centric: In this class, the program control can be transferred from a host to using code mobility. Agent-based platforms for WSNs belong to this class. Agents are interpreted programs that can migrate from one host to another during its lifetime. Application modules are

presented to the network as autonomous mobile agents. This approach supports the dynamic evolution of applications, although it suffers from high overhead making it less suitable for resource constraint environment. Agilla [202] and Impala [203] are examples of agent-based platforms for WSNs.

4.3.2.2. MODULAR

In this class, programs are constructed from smaller standardized units called modules that are linked together. The decomposition of the program structure into modules improves flexibility, reusability, and maintainable of software. The modular programs can be further divided into two subclasses based on the time modules are linked..

1) Statically Linked: In this class, modules are linked at system integration time. Once integrated, the program structure cannot be altered. At runtime, statically linked modular program have similar characteristic to monolithic programs, however they allow more flexibility at development and maintenance times. In WSN, applications built for TinyOS belong to this category.
2) Dynamically Linked: In this class, modules are linked at runtime. Programs can change the bindings between modules when needed. This enables structural plasticity of programs and minimizes the overhead of software updates. Modular program with dynamic control can be further classified into two classes according to the scope of linking.
    i. Host Centric: In this class of dynamically linked modular programs, all program modules must exist on the same host (NSE node). In WSNs, applications built for RETOS ([204] refer for details on this OS) belong to this category.
    ii. Network Centric: In this class of dynamically linked modular programs, different modules composing a program could exist at different hosts. This allows programs to scale beyond the storage and computational capabilities of a single host of networked sensor environments node.

The CPS eventually eCPS platform, presented in this dissertation, is based on the programming class that are Modular_Dynamically Linked_Network Centric, which also meet the requirements that are presented in section 3.3.4. More precisely in the scope of this research work Java has been selected as the programming language for implementation (refer [205] *and* [206] *for detailed understanding of modular, dynamic linked and network centric programming features and methodologies in Java).*

5. SUMMARY OF LITERATURE REVIEW

There are two ways to write error-free programs; only the third one works.

-Alan J. Perlis

This chapter provides the summary of the literature review and builds up a common understanding of important concepts which is important for understanding the following chapters. In the beginning of this chapter is presented the definitions of concepts that have lead towards the research results and hence this dissertation. The following section makes the analysis of the important literatures in different domains with focus on their main contributions and identified gaps. By the end of this chapter it is expected that the reader will have understanding of terms and concepts widely used in this dissertation and also have the idea about the state-of-art on different domains which have important impact in this research work.

5.1. ALIGNMENT OF CONCEPTS

In this section we will present some of the important concepts that will be widely used in this dissertation. The main purpose of this section is to provide a common understanding of the terms.

- **Definition 5.1:** ***Evolution***: Evolution is the process by which changes happen over time governed by the environmental factors.
- **Definition 5.2:** ***Evolutionary Algorithm***: An algorithm which incorporates aspects of natural selection or survival of the fittest in order to optimize the given problem. An evolutionary algorithm maintains a population of structures that evolves according to rules of selection, recombination, mutation and survival, referred to as genetic operators. A shared environment determines the fitness or performance of each individual in the population. The fittest individuals are more likely to be selected for reproduction (retention or duplication), while recombination and mutation modifies those individuals, yielding potentially superior ones.
- **Definition 5.3:** ***Fitness function***: A fitness function is a particular type of objective function that is used to summarize, as a single figure of merit, how close a given design solution is to achieving the set aims. In particular, in the fields of genetic programming and genetic algorithms, each design solution is commonly represented as a string of numbers (referred to as a chromosome). Each design solution, therefore, needs to be awarded a figure of merit, to indicate how close it came to meeting the overall specification, and this is generated by applying the fitness function to the test, or simulation, results obtained from that solution.
- **Definition 5.4:** ***Emergent Behaviors***: Emergent behaviors or emergent properties are the new behaviors which can appear when a number of simple

entities (agents) operate in an environment, forming more complex behaviors as a collective.

- **Definition 5.5:** ***Evolutionary System***: The computational system that undergoes evolution over time due to the impact of the changes in the system run-time environment.
- **Definition 5.6:** ***Genotype***: Genotype refers to the sequence of information encoded that defines the particular a specific characteristic (phenotype) of that cell (or sub-system in the context of this thesis).
- **Definition 5.7:** ***Phenotype***: A phenotype is the composite of an organism's (sub-system in the context of this thesis) observable characteristics or traits, such as its morphology, development, behavior, and products of behavior. A phenotype results from the expression of a system's genes as well as the influence of environmental factors and the interactions between the two.
- **Definition 5.8:** ***Behavioral Modelling***: Modelling of the system so that the system acts and responds as per the specified role under similar situations. It is important to state that behavioral modelling is used to examine systems of behavior, rather than the behavior of an individual at any particular time.
- **Definition 5.9:** ***Sensorial Modelling***: Modelling of the system with group of sub-systems that can be used for detecting and understanding the world around. This also includes modelling of actions or behaviors that are intrinsic to particular type of information detected.
- **Definition 5.10:** ***Abstract Machine***: An abstract machine is a model of a computer system (considered either as hardware or software) constructed to allow a detailed and precise analysis of how the computer system works. Such a model usually consists of input, output, and operations that can be performed (the operation set), and so can be thought of as a processor. Turing machines are the best known abstract machines.
- **Definition 5.11:** ***Knowledgebase***: A collection of data organized in a form that facilitates analysis by automated deductive processes, consisting of concepts, data, objectives, requirements, rules, and specifications
- **Definition 5.12:** ***Ontology***: Ontology is a formal naming and definition of the types, properties, and interrelationships of the entities that really or fundamentally exist for a particular domain of discourse. Ontology compartmentalizes the variables needed for some set of computations and establishes the relationships between them.
- **Definition 5.13:** ***Cyber Physical Systems***: A cyber physical system (CPS) are systems with deeply intertwined physical and software components operating on different spatial and temporal scales, exhibiting multiple and

distinct behavioural modalities, and interacting with each other in a myriad of ways that change with context.

- **Definition 5.14:** ***Evolutionary Cyber Physical Systems***: The class of CPSs those are capable of detecting and enacting emergent behaviors leading towards system evolution over a long discourse of time.

5.2. SYNTHESIS

In many literatures and reports CPSs have been identified as the core enabling and disruptive technology for securing economic leadership in embedded systems/ICT, with enormous social and economic importance. Thus mastering the engineering of complex and trustworthy CPS is important for allowing our industries to implement CPS-based business models. This PhD work has aimed high to study evolutionary models for CPS, inspired by theory of natural evolution. The major foundation for this research work is based on the prior-knowledge in three major fields as shown in Figure 5-1.

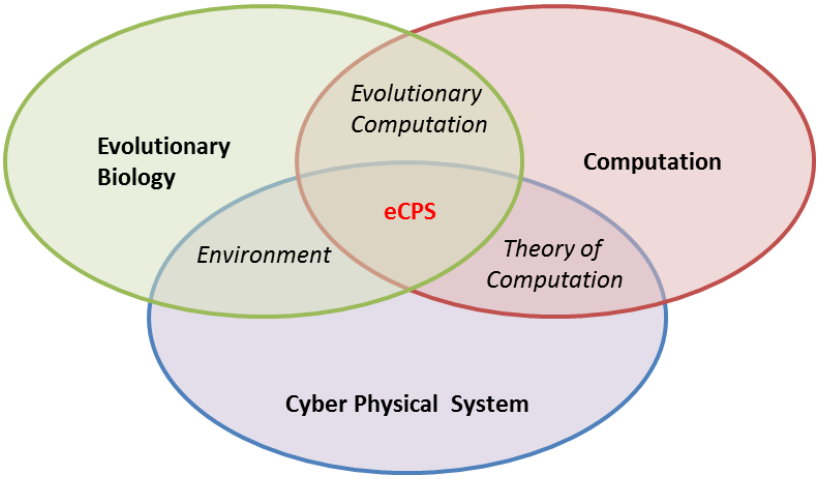


Figure 5-1 Major foundations of eCPS

The overall studies made in the course of this dissertation have been motivated by the previous research works in the domain of evolutionary biology, computation and cyber physical systems. During the study of various literatures, it was quite evident observation on overlaps between these diverse scientific domains. The major overlap between the evolutionary biology and CPS is the factor "***Environment***". While lots of studies have been made in biology on the impact of environment for evolution of organisms, there has been relatively less study on how the environment can be modelled to be utilized by the computational systems. This was an important aspect of study during this research work. At the same time the methodology for defining evolutionary strategies has been adopted from the research results from evolutionary biology which has been well formulated and has been applied in computational systems. This cross domain research has lead towards the research domain of "***Evolutionary Computation***", which is a clear overlap between

the two scientific domains of evolutionary biology and computation. CPSs are computational systems thus obviously share the overlap with the computation and the one of main interest in this dissertation has been "***Theory of Computation***". Theory of computation is a branch of theoretical computer science and mathematics, which deals with how efficiently problems can be solved on a model of computation. The major interest for us in the scope of this dissertation is formulation of generic model of computation for CPS by considering both the physical and cyber world.

The literature review was performed based on the synthesis made above- with more alignment towards the computational systems. The literature review thus, performed is summarized in Figure 5-2.

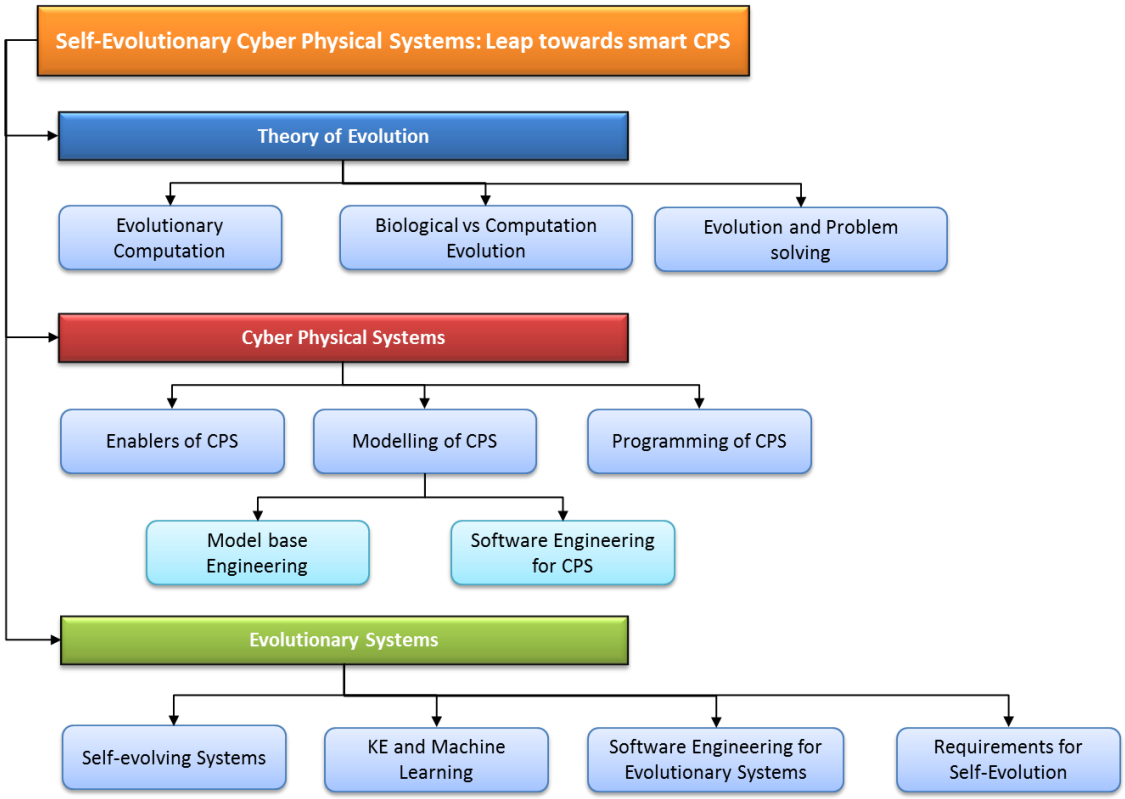


Figure 5-2 Literature Review summary with major research areas and sub-areas

Thus performed literature review has provided a solid base for the research work that has been explained in this dissertation. In the next section is provided the summary of some of the important references.

5.2.1. *SUMMARY OF IMPORTANT LITERATURES*

Table 5-1 presents the summary of some of the important literatures that have been analysed to understand the state of the art in different domains.

Table 5-1 Some important literatures

Domain	Important References	Important Keywords	Year
Evolutionary Computation	Evolution strategies – A comprehensive introduction [207]	***computational intelligence; design principles for genetic operators; evolutionary computation; evolution strategies, optimization***	2002
	Proposal of a toolset for the improvement of industrial systems' lifecycle sustainability through the utilization of ICT technologies [108]	***Industrial systems; Life Cycle Assessment; Life Cycle Costing; Optimization; Product lifecycle; Sustainability***	2015
	Applying Digital Evolution to the Design of Self-Adaptive Software [135]	***digital evolution; evolutionary computation; autonomic computing***	2009
	Knowledge Extraction based on Evolutionary Learning (KEEL): Analysis of Development Method, Genetic Fuzzy System [145]	***Data mining; Evolutionary algorithms, Genetic Fuzzy System***	2012
	Variants of Evolutionary Algorithms for Real-World Applications [54]	***Differential Evolution; Estimation of Distribution Algorithms; Co-evolutionary Algorithms and Multi-Objective Evolutionary Algorithms***	2012
	Co-evolutionary and genetic algorithm based building spatial and structural design [57]	***Co-evolutionary Design; Genetic Algorithm; Topology Optimization.***	2015
	Analyzing Evolutionary Algorithms: The Computer Science Perspective [64]	***Black-Box Optimization; Evolutionary Computing (EC); Evolutionary Algorithms (EAs); Natural Computing;***	2013
Cyber Physical Systems	Cyber-Physical Systems: A New Frontier [124]	***Sensor Networks; Ubiquitous; Trustworthy Computing***	2009
	Modelling cyber–physical systems [155]	***Analytical models; Computational modelling; Hybrid intelligent systems; Mathematical model;***	2012
	Analysis of the State-of-the-Art and Future Challenges in Cyber-physical Systems of Systems [8]	***Emergent Behaviour; Dynamic Reconfiguration; Continuous Evolution***	2015
	Considering cognitive aspects in designing cyber-physical systems: an emerging need for transdisciplinary [138]	***cyber physical systems; information-intensive products***	2013
	A model-based design methodology for cyber-physical systems [156]	***cyber-physical systems; embedded systems; model-based design***	2011
	Cyber-Physical Systems-Concept, Challenges and Research Areas [127]	***architecture; cyber-physical systems; dependability; design; modelling***	2012

	Smart cyber society: Integration of capillary devices with high usability based on Cyber–Physical System[128]	***Cyber–Physical Systems; Cyber society; Virtual communication platform; Web of things (WoT)***	2016
	A Cyber Physical Interface for Automation Systems—Methodology and Examples [131]	***adaptive algorithms; ball screw; cyber physical system; health monitoring***	2015
	Cyber-Physical System Components Composition Analysis and Formal Verification Based on Service-Oriented Architecture [187]	***Cyber-Physical system; Service Composition; Service-Oriented Architecture; Time-Space pi-calculus***	2012
	Robust Cyber–Physical Systems: Concept, models, and implementation [151]	***Cyber–Physical Systems; Stability; Security; Sensors and actuators***	2016
Self-Evolutionary Systems	Self-organization and Specialization in Multiagent Systems Through Open-ended Natural Evolution [143]	***Open-ended natural evolution; multi-agent systems; self-organization***	2012
	On the Role of Embodiment for Self-Organizing Robots: Behavior As Broken Symmetry [90]	***self-organization; humanoid; Unsupervised learning***	2014
	Computational Intelligence: An Introduction [67]	***Multi-Layer Perceptrons; Self-organizing Maps; Behavioural Simulation***	2013
	Designing Self-Adaptive Embedded Real-time Software - Towards System Engineering of Self-Adaptation [107]	***Self-adapting Software; Artificial Immune Systems; Danger Theory; Mixed-criticality Systems***	2014
	From evolving software towards models of dynamically self-assembling processing systems [102]	***self-replication; evolving software system; dynamical assembled programming***	2006
	Anomaly Detection and its Adaptation: Studies on Cyber-Physical Systems [101]	***machine learning; data mining; Supervisory Control and Data Acquisition; Mobile Ad hoc Network***	2013
	Considering cognitive aspects in designing cyber-physical systems: an emerging need for transdisciplinarity[138]	***cognitive engineering, mental models, cognitive architectures***	2013
	Autonomous Perception and Decision-making in cyber-physical Systems[84]	***Science of Autonomy, Cyber Systems, Information Technology, Decision & Control***	2013

SECTION III: THEORETICAL AND TECHNOLOGICAL RESULTS

In this section of the dissertation is presented the scientific and technological results of the research work with necessary theoretical and methodological foundation leading towards evolutionary cyber physical systems. Within these section principles from the theory of computation more precisely model of computation has been applied to form the necessary model for evolution, cyber physical systems and eventually evolutionary cyber physical systems. Methodological and technical foundations are also presented in this section that forms the base for implementation and validation of scenarios. This section is divided into two chapters: ***Chapter 6: Towards self-Evolutionary Cyber Physical Systems*** *and* ***Chapter 7: Implementation and Validation***

Chapter 6: *This is an important chapter which has been divided into sections building up the theoretical and methodological foundation. In order to perform a rigorous study of computation, this chapter works with mathematical abstraction of computers called a model of computation. The base for this work is Turing Machine and automata because they are considered the most powerful possible reasonable model of computation. The first section i.e. 6.1 presents the necessary theoretical model for defining CPS along with modelling of behavioural and sensorial model necessary for CPS. The second half of the chapter i.e. 6.2 provides the methodological and technical foundations for the realization of eCPS. In the methodological foundation is provided the necessary methodologies for realization of eCPS including the life cycle and necessary design methodology for realization of CPS and eventually eCPS. The technical foundation provides the technical perspective supported by various facets of technical specification. This chapter starts with presenting the reference architecture that captures different layers and components necessary for realization with necessary details specification of different layers. The following section presents the technology blocks that are applicable at different layers and for various purposes to realize a complete eCPS platform. This chapter also presents dynamic deployment framework, which is an important necessity for run-time environment of eCPS.*

Chapter 7: *In this chapter is presented the scenarios that have been implemented and used for validation of the theories and principles presented in the preceding chapters. The scenarios that have are presented in this chapter provide interesting research results that have been one of the core contribution of this dissertation. At the same time this chapter provides practical understanding of the overall research work.*

On the whole by the end of this section, it is expected that for the reader will have a clear understanding of the important theoretical, technical and practical contributions of this dissertation.

6. TOWARDS SELF-EVOLUTIONARY CYBER PHYSICAL SYSTEMS

Everybody gets so much information all day long that they lose their common sense.

- Gertrude Stein

This chapter is an important chapter for theoretical and methodological understanding of self-evolutionary cyber physical systems. The field of evolutionary computation has been a research work that has itself been an evolving community of people, ideas, and applications. Although one can trace its genealogical roots as far back as the 1930s, it was the emergence of relatively inexpensive digital computing technology in the 1960s that served as an important catalyst for the field. Firstly in this chapter is presented the computational model of evolutionary machines, cyber physical systems and eventually evolutionary cyber physical systems. At the same time it provides formal modal for representing behavioural and sensorial model of CPS, which are deemed important for complete modelling of CPS. The second half of the chapter provides necessary methodological foundations which form a strong base for the following sections. By the end of the chapter, the reader is expected to be familiar with computational model of evolutionary machines, CPS and clearly understand the methodologies that will be used in following chapters for practical technical implementation.

Living organisms exhibit an amazing ability to adapt to a changing environment, both in the short term (phenotypic plasticity) and in the longer term (Darwinian evolution). Hence, to design dynamic and resilient computational systems, researchers often take inspiration from nature. Moreover, many organisms exhibit traits that are desirable in self-managing computing systems (which were mentioned as requirements in section 3.3.4): system monitoring (senses, awareness); short-term changes in priorities (reflexes, sleep); system reconfiguration (muscle growth, calluses); self-repair (blood clotting, tissue healing); intrusion detection/elimination (immune systems). For this reason, many researchers turn to biologically-inspired methods to construct highly adaptive systems [208], [209] both by mimicking the designs produced by nature, and by evolving new ones. Evolutionary computation abstracts the evolutionary process and applies it in a purely algorithmic form to problem solving. An, very interesting work towards building robust, a flexible computational system is digital evolution [210]. In this method, a population of self-replicating computer programs exists in a user-defined computational environment and is subject to mutations and natural selection. These “digital organisms” are provided with limited resources whose use must be carefully balanced if they are to survive. Over generations, these organisms evolve to optimize resource usage and thrive if they are able.

In this chapter we will provide the foundation that will lead towards the realization of self-evolutionary cyber physical systems.

6.1. THEORETICAL FOUNDATION

Foundation for evolutionary computing dates back to the time of Alan Turing, who is considered as one of the founders of theoretical computer science. Turing’s basic model

of computation- Turing machine has been the base of computer science and engineering. Besides the Turing machine, during his work at National Physical Laboratory, Turing worked on the Automatic Computing Engine (ACE) under the supervision of Sir Charles Darvin (the grandson of the founder of the theory of evolution). Before leaving for Manchester in 1948, Turing produced a final report on ACE which can be viewed as a blueprint for the future field of evolutionary computation titled "Intelligent Machinery (Turing 1948)", this report was left unpublished until 1968. In this report, among other futuristic ideas, including robots taking country walks, Turing proposed new models of computation, which he called unorganized machines (***u-machines***) [211]. There were two types of u-machines, those based on Boolean networks and those based on finite state machines. Turing took his inspiration from the working of the human cortex, and its ability for self-adaptation.

- **A-type** and **B-type u-machines** were Boolean networks made up of a fixed number of two-input NAND gates (neurons) and synchronized by global clock. While in A-type u-machines the connections between neurons were fixed, B-type u-machines had modifiable switch type interconnections. Starting from the initial random configuration and applying a kind of genetic algorithm, B-type u-machines were supposed to learn which of their connections should be on and which off.
- **P-type u-machines** were tapeless *Turing Machines* reduced to their *Finite State Machine control*, with an incomplete transition table, and two input lines for interaction: the pleasure and the pain signals. For configurations with missing transitions, the tentative transition to another state could be reinforced by "pleasure" input from the environment, or cancelled in the presence of "pain".

In his B-type u-machines, Turing pioneered two areas at the same time: ***neural networks and evolutionary computation (more precisely, evolutionary artificial neural networks EANNs)***, while his P-type u-machines represent ***reinforcement learning***. However, this work had no impact on these fields, due to the unfortunate combination of Turing's death and the twenty-year delay in publication (for more details follow [212,213]). But Turing's work forms the basis for theoretical foundation that builds up towards the formal representation of evolutionary cyber physical systems in the following subsections.

6.1.1. *BASIC EVOLUTIONARY MACHINES*

In this section we will provide the theories on evolutionary machines based on Turing's works and some recent literatures such as [211,214,215]. In technical terms, an evolutionary algorithm is a probabilistic beam hill climbing search algorithm directed by the chosen fitness function. It means that the beam (population size) maintains multiple search points, hill climbing implies that only a current search point from the search tree is remembered and used for optimization (going to the top of the hill), and the termination condition very often is set to the optimum of the fitness function. Let X be the

representation space, also called the optimization space, for species (systems) used in the process of optimization and a fitness function $f: X \to \boldsymbol{R}$ is chosen.

Definition 6.1. A generic evolutionary algorithm (EA) *E* can be represented as the collection *E = (X, X[0], F, f, s, v, R)* and described in the form of the functional equation (recurrence relation) R working in a simple iterative loop in discrete time t, defining generations *X[t] , t = 0, 1, 2, 3, ...:*

$$X[t+1] = s\,(v\,(X[t]))$$

Where,

- *X* is representation space; (e.g., *X* consists of fixed binary strings for genetic algorithms (GAs), of Finite State Machine descriptions for evolutionary programming (EP), of parse trees for genetic programming (GP), of vectors of real numbers for evolution strategies (ES));
- *s* is selection operators (e.g., *truncation*, *proportional selection* or *tournament*)
- *v* is variation operators (e.g., *mutation*, *crossover* or some *combination of mutations and crossover*)
- *f* is a fitness function s.t. $f: X \to R$, which typically takes values in the domain of nonnegative real numbers and is extended to the subsets of the set *X* by the following rule if $Y \subseteq X$, then $f(Y) = max\,\{f(x);\ x \in Y\}$
- *C* is termination or search condition (goal of evolution);
- *X[0]* is an initial population;
- $X[t] \subseteq X$ is the population produced on the (*n-1*)-th stage of the evolutionary algorithm (EA) *A*;
- $F \subseteq X$ is the set of final populations satisfying the termination condition (goal of evolution).

Thus, the dynamics of the evolutionary algorithm *A* is described in the form of the functional equation (recurrence relation) working in a simple iterative loop with parts of the space *X* called generations in discrete time *t = 0,1,2,3,...*, which has been widely discussed in [216] and [217]. This functional equation describes how the evolutionary algorithm A taking the generation i produces the generation $X[t+1] \subseteq X$.

Definition 6.1 is applicable to all typical EAs, including GA, EP, ES, GP.

Now, we define a formal algorithmic model of evolutionary computation - an evolutionary automaton also called an evolutionary machine. Let K be a class of automata.

Definition 6.2. A *basic evolutionary **K**-machine* (*BEM*), also called *basic evolutionary **K**-automaton*, is a (possibly infinite) sequence $E = \{E[t]; t = 0, 1, 2, 3, ... \}$ of automata $E[t]$ from **K** each working on the population $X[t]$ $(t = 0, 1, 2, 3, ...)$ where:

- the automaton $E[t]$ called a component, or more exactly, a level automaton, of E represents (encodes) a one-level evolutionary algorithm that works with the generation $X[t]$ of the population by applying the variation operators v and selection operator s;
- the first generation $X[0]$ is given as input to E and is processed by the automaton $E[0]$, which generates/produces the first generation $X[0]$ as its output, which goes to the automaton $E[1]$;
- for all $t = 1, 2, 3, ...$, the generation $X[t + 1]$ is obtained by applying the variation operator v and selection operator s to the generation $X[t]$ and these operations are performed by the automaton $E[t]$, which receives $X[t]$ as its input;
- the goal of the BEM E is to build a population Z satisfying the search condition.

An important property of living systems is their ability to ***change in the process of functioning***. To reflect this property, we introduce reconfigurable evolutionary K-machines. This model of evolutionary computation is rooted in reflexive Turing machines introduced as a generic model for programs (algorithms) that change (improve) themselves while they such as reconfigurable software [218] and reconfigurable and transformable computers [219–221].

Definition 6.3. A *basic reconfigurable evolutionary **K**-machine* (BRCEM) is a basic evolutionary **K**-machine $E = \{E[t]; t = 0, 1, 2, 3, ... \}$ in which it is possible to change (transform) the automata $E[t]$ in the process of computation.

A new direction in computer technology is based on the principles of reconfigurable computer. In contrast to conventional computers, a reconfigurable computer computes a function by configuring functional units and wiring them up in space. This allows, for example, parallel computation of specific, configured operations. A reconfigurable computer can be easily and quickly modified from a remote location to upgrade its performance or even to perform a completely different function. As a result of such advantages, reconfigurable computers serve as powerful tools for many applications, such as research and development tools for sophisticated electronic systems or verification on electronic designs.

Another important class of evolutionary machines is evolutionary finite automata [222].

Definition 6.4. An *evolutionary finite automaton (EFA)* is an evolutionary machine E in which all automata $E[t]$ are finite automata $G[t]$ each working on the population $X[t]$ in generations $t = 0, 1, 2, 3, ...$

We denote the class of all evolutionary finite automata by *EFA*.

It is possible to consider deterministic finite automata, which form the class *DFA*, and nondeterministic finite automata, which form the class *NFA*. This gives us two classes of evolutionary finite automata: *EDFA* of all deterministic evolutionary finite automata and *ENFA* of all nondeterministic evolutionary finite automata. Note that it is also possible to consider reconfigurable evolutionary finite automata. Evolutionary Turing machines [223], are another important class of evolutionary machines.

Definition 6.5. An *Evolutionary Turing Machine* (*ETM*) $E = \{TM[t]; t = 0, 1, 2, 3, ...\}$ is an evolutionary machine E in which all automata $E[t]$ are Turing machines $TM[t]$ each working on population $X[t]$ in generations $t = 0, 1, 2, 3, ...$

Note that similarly, it is also possible to consider *reconfigurable evolutionary Turing machines*.

Variation and selection operators are recursive to allow problem computation on Turing machines. So, it is natural to assume that the same Turing machine computes values of the fitness function f. This brings us to the concepts of weighted Turing machines and weighted evolutionary Turing machines, which were introduced and studied in [224].

6.1.1.1. COMPUTATIONS BY EVOLUTIONARY MACHINES

According to the theory of algorithms and computation ([225], [226]), there are three basic types of automaton functioning:

- *Computing type* of functioning is when the automaton receives an input and gives an output. Automata working in the computing manner are called transducers.
- *Accepting type* of functioning is when the automaton receives an input and either accepts this input or does not accept it. Automata working only in the accepting manner are called acceptors.
- *Generating type* of functioning is when the automaton does not receive an input but gives an output. Automata working only in the generating manner are called generators.

Note that acceptors can also give some output although their result is either acceptance or rejection, i.e., the result and output are not the same for acceptors. Evolutionary machines consist of components called level automata. This means that there are local and global modes of evolutionary machines functioning, i.e., functioning of each level automaton in the evolutionary machine goes according to the local mode, while functioning of the whole evolutionary machine goes according to the global mode [211].

When all automata in a class K are transducers or generators, they give output. In a general case, this output consists of two parts: transaction output and terminal output.

Definition 6.6. Transaction output of the level automaton *E[t]* is the generation *X[t]*, which is transmitted to the next level automaton *E[t + 1]*.

This means that the transaction output always remains in the evolutionary machine, providing interaction of the components.

Definition 6.7. Terminal output of the level automaton *E[t]* is given for some external system, e.g., for the user.

For instance, the level automaton *E[t]* can inform the user about the maximal or minimal value of the fitness function *f(x)* for the generation *X[t]*, i.e., the optimum of the fitness performance measure *f(x[t])* of the best individual from the population *X[t]*.

Note that to work in the computing manner, an evolutionary machine has to give some terminal outputs. At first, let us we consider the following global accepting modes of evolutionary automaton functioning:

1. The existential mode is characterized by the rule: An evolutionary automaton E accepts the generation *X[0]*, e.g., in the form of a word w, given to the level automaton *E[0]* as input if and only if there is a level automaton E[t] that accepts the generation *X[t - 1]* (which can be also in the form of a word) produced by the level automaton *E[t - 1]*.
2. The coexistential mode is characterized by the rule: An evolutionary automaton E rejects the generation X[0], e.g., in the form of a word w, given to the level automaton E[0] as input if and only if there is a level automaton E[t] that rejects the generation X[t - 1] (which can be also in the form of a word) produced by the level automaton E[t - 1].
3. The universal mode is characterized by the rule: An evolutionary automaton E accepts the generation *X[0]*, e.g., in the form of a word w, given to the level automaton *E[0]* as input if and only if all level automata *E[t]* accept the corresponding generation *X[t - 1]* (which can be also in the form of a word) produced by the level automaton *E[t - 1]*.
4. The infinitary mode is characterized by the rule: An evolutionary automaton *E* accepts the generation *X[0]* given to the level automaton *E[0]* as input if and only if there are infinitely many level automata *E[t]* each of which accepts the generation *X[t - 1]* produced by the level automaton *E[t - 1]*.

Till this point it has been clearly defined the way evolutionary machines can be defined and the way they function. But in the scope of this research work, the objective is the study of CPSs, which are hybrid system i.e. digital real-time systems embedded in analog environments. For this purpose hybrid automata are used to model such hybrid systems as adapted from [227]. Following section explains the formal modelling of CPS and eventually eCPS.

A dynamical system is simply a system whose "state" evolves with "time" governed by a fixed set of rules or "dynamics". The state of a dynamical system is specified as valuations of the variables of interest in the system. Depending upon the nature of variables (discrete or continuous) and the notion of time (discrete or continuous) the dynamics of variables can be specified by differential equations or discrete assignments. For the purpose of this research work, dynamic systems are classified into three broad classes:

i. discrete systems where both the notion of time and the variables are discrete,

ii. continuous systems where the notion of time is continuous, while the variables are continuous, and

iii. hybrid systems where some variables are continuous and some are discrete, and although the notion of time is continuous, special dynamic-changing events can happen at discrete instants. Notice that both discrete and continuous systems can be considered as subclasses of hybrid systems.

CPS are hybrid systems and hybrid automata provide an intuitive and semantically unambiguous way to model cyber physical systems, and a number of case-studies [228–231] demonstrate their application for the analysis of cyber physical systems. Hybrid systems share their properties with both discrete as well as continuous systems, as their state progresses with time in both discrete jumps as well as continuous flows. In this section we present hybrid automata, a combination of extended finite state machines and continuous dynamical systems, where in every control mode the dynamics of the variables of the system can be specified using ordinary differential equations (ODEs).

Definition 6.8. (*Hybrid Automata*: Syntax):

A hybrid automaton is a tuple $H = \langle M, M_0, \Sigma, X, \Delta, I, F, V_0 \rangle$ where:

- M is a finite set of control modes including a distinguished initial set of control modes $M_0 \subseteq M$,
- Σ is a finite set of actions,
- X is a finite set of real-valued variable,
- $\Delta \subseteq M \times pred(X) \times \Sigma \times pred(X \cup X') \times M$ is the transition relation,
- $I : M \rightarrow pred(X)$ is the mode-invariant function,
- $F : M \rightarrow (R^{|X|} \rightarrow R^{|X|})$ is the mode-dependent flow function characterizing the flow for each mode $m \in M$ as the set of ODEs $\dot{X} = F(m)(X)$, and
- $V_0 \in pred(X)$ is the set of initial valuations.

The execution of a hybrid automaton begins in an initial configuration (m_0, v_0) where $m_0 \in M_0$ is an initial mode and $v_0 \in V_0$ is an initial valuation satisfying $[\![v_0 \in I(m_0)]\!]$. The

system stays in a mode for some time, say $t_1 \in \mathbb{R}\geq 0$, and while the system stays in a control mode m the valuation of the variables changes according to ODE specified by the flow *F(m)* of the corresponding mode. After spending $t_1 \in \mathbb{R}\geq 0$time in mode m_0 an enabled transition (m_0, g, a, j, m_1) is non-deterministically chosen and executed.

Definition 6.9. (*Hybrid Automata: Semantics*): The semantics of a hybrid automaton $H = <M, M_0, \Sigma, X, \Delta, I, F, V_0>$ is given as a state transition graph T^H s.t.

$T^H=(S^H, S^H_0, \Sigma^H, \Delta^H)$

where:

- $S^H \subseteq (M \times \mathbb{R}^{|X|})$ is the set of configurations of *H* such that for all $(m, v) \in S^H$ we have that $v \in [\![I(m)]\!]$;
- $S^H_0 \subseteq S^H$ s.t. $(m, v) \in S^H_0$ if $m \in M_0$ and $v \in V_0$;
- $\Sigma^H = \mathbb{R} \geq 0 \times \Sigma$ is the set of labels;
- $\Delta^H \subseteq S^H \times \Sigma^H \times S^H$ is the set of transitions such that $((m,v),(t,a),(m', v')) \in \Delta^H$ if there exists a transition $\delta = (m, g, a, j, m') \in \Delta$ such that
  - $(v \oplus F(m)^t) \in [\![g]\!]$;
  - $(v \oplus F(m)^\tau) \in [\![I(m)]\!]$ for all $\tau \in [0, t]$;
  - $v' \in (v \oplus F(m)\ t)[j]$; and
  - $v' \in [\![I(m')]\!]$.

For understanding the operator $\oplus$ consider a configuration of a hybrid automaton is a tuple (m, v) where $m \in M$ is a mode and $v \in \mathbb{R}^{|X|}$ is a variable valuation. Then for a Lipschitz continuous flow function $F : M \rightarrow (\mathbb{R}^X \rightarrow \mathbb{R}^{|X|})$, a valuation $v \in \mathbb{R}^{|X|}$, a mode $m \in M$, and a time delay $t \in \mathbb{R} \geq 0$ we define $(v \oplus F(m)^t)$ for the unique valuation *f(t)* where *f* is the unique run of the continuous dynamical system $(X, F(m), v)$. For a jump predicate $j \in pred(X \cup X')$ and valuation v we define *v[j]* for the set of valuations $v' \in \mathbb{R}^{|X|}_{\geq 0}$ such that $(v, v') \in j$.

Follow [232] *for detailed understanding of State Transition Diagrams and* [233] *for understanding Lipschitz continuous flow.*

Now finally with the definition of *Hybrid Automata* let's define the evolutionary hybrid automaton which will be the basic model for eCPS.

Definition 6.10. An *evolutionary hybrid finite automaton (EHFA)* is an evolutionary machine *E* in which all automata *EH[t]* are finite hybrid automata H*[t]* each working on the population *X[t]* in generations *t = 0, 1, 2, 3, ...*

We denote the class of all evolutionary finite automata by *EHFA*. These classes of *EHFA* are the basic model that represents evolutionary cyber physical systems.

In the following sections we will present two important aspects of CPS i.e. behavioural model and sensorial model that will provide further details for *EHFA.*

6.1.3. *BEHAVIOURAL MODELLING FOR CPS*

It has been a clear and important functionality for all CPSs that they have to deal with inherent dynamic nature of environments. In dealing with the dynamic nature of CPS, most of the recent research work have either ignored the temporal complexities in CPS or address them in an ad hoc manner [234]. However, in real world scenario, we cannot expect the same set of conditions to hold forever, and need explicit mechanisms to model changes and corresponding responses. This can be achieved by following the methodology for event detection, identification and feedback control loops, which have been independently studies in some other fields like computer vision, control systems etc. [235], [236], [237]. But, the main short coming of these works is that they do not consider what to do next with these events and typically do not use formal temporal logic to represent these events or handle actuator response. This becomes a necessity to completely model CPS or more precisely an important consideration for eCPS.

In order to accommodate this requirement it's necessary to incorporate event calculus and situational calculus in the design process of CPS. Work in areas like event calculus and situation calculus [238–240] on the other hand, provide valuable formal models for representing events and situations based on their impact on the real world. Also, research in discrete event control [241] has provided some very useful tools for handling event based input and output in generic cyber physical settings. But to solve control theoretic problems for powerful cyber physical applications requires general purpose tools which consider event detection, formally study their impact on world models and provide robust tools for deciding control actions. Hence, it's necessary to formulate behavioural modelling methodology for control mechanism in CPS. Such a control approach should support

i. The ability of the controller to reason based on symbols (rather than just signals)
ii. Inherent support for temporal reasoning, and
iii. Explicit inclusion of domain semantics.

In the contextual environment of CPS, the dynamic environment is composed of two parts viz. system-controllable (where the actions are totally controlled by the system) and system uncontrollable (where the actions are exogenous to the system e.g. user actions). To model worlds of CPS, fluent calculus is very useful and is followed in the scope of this work. Fluent Calculus [242], is a formalism for expressing dynamical domains in first-order logic. Fluent calculus has fluents, which are first order predicates having an argument that depends on time. For example the predicate *On(box,table)* cannot be always true (or false). We must reify the temporal aspect of this predicate as a variable to

make it more useful e.g. using situation '*s*' to get *On(box,table, s)*. Thus, fluents can be used to define control actions for CPS based on some situation. All the actions impact the fluent values and can potentially cause the system to move away from its equilibrium state. The system then undertakes the control actions to move back towards equilibrium. The control actions thus undertaken can further change the system state, resulting in respective control actions. Such a process continues until the system return to equilibrium state, at which it waits for next user action.

In CPSs domain, the actions are captured by the system using sensors, and the control actions are undertaken using available actuators. An important point to note in this discussion is that the control action is determined by the 'state' variable, which elegantly combines the user-action input with the previous state value. This resultant state which is the necessary and sufficient world descriptor to decide the control output is defined as situation in our system. This methodology of working is exactly same as the working of hybrid automata (i.e. the primitive model of all CPS) as discussed in section 6.1.2.

6.1.3.1. SITUATION BASED CONTROL MODEL

As, discussed in the previous section, it is clear that CPS required methodology for defining control actions based on situation. Situational calculus thus becomes handy for behavioural modelling of CPS. Situation calculus pioneered by McCarthy [243] and subsequently by Reiter [238], [244] and has been successfully employed to logic, robotics [245] and is starting to be used in databases. But it has not been applied to CPSs.

Situation Calculus is a logic formalism designed for representing and reasoning about dynamical domains [246]. It builds upon traditional predicate, 1st and 2nd order calculus, but is different because it allows for truth values to change over time. The main use of Situation Calculus (on similar lines to traditional calculus), is in using a set of truth values (facts) about a closed world and some reasoning mechanisms (predicates or functions), to infer new truth values which have not explicitly been provided earlier. Let's say $L_{sitCalc}$ is a second order language with equality and has 4 disjoint components.

1. Actions (*A*) for actions i.e. those which change the 'state' of the world,
2. Situations (*S*) for 'history of events'
3. Objects (*O*) as the default sort for everything else.
4. The most important component of Situation Calculus is the fluent (*F*) sort, which defines truth statements which are dependent on the situation. Fluents can be relational (typically give True/False answers) or functional (return any value as computed).

Based on these components of situational calculus, the behavioural calculus for CPS can be defined as below.

Definition 6.11. The language for CPS behavioural calculus can be defined as: $\Sigma = <A, S, O, F>$ where A, S, O, F are as defined above.

With this definition, within Σ we can formulate action theories that describe how the world changes as a result of the available actions, based on the action theories as defined in [238]. An action theory D has the following form:

$$D = \Sigma \cup Df \cup D_u \cup D_a \cup D_s \cup D_0 \text{ where,}$$

- Σ is set of foundational axioms of situation as described before.
- D_f consists of the axioms for equality and a set of domain independent foundational axioms which formally define legal situations including $(do(a, s) = do(a', s')) \subset (a = a' \wedge s = s')$
- D_u is the set of unique-names axioms for actions. $A(x) = A'(y) \wedge A(x) = A(y) \supset x = y$
- D_a is a set of action precondition axioms, one per action symbol A, of the form $Poss(A(y), s) \equiv \Pi A(y, s)$.
- D_s is a set of successor state axioms (*SSAs*), one for each fluent symbol F, which characterizes all the ways the value of a particular fluent can be changed such that $Poss(a, s) \rightarrow [F(\bar{x}, do(a, s)) \leftrightarrow \gamma^+_F(\bar{x}, a, s) \vee (F(\bar{x}, s) \wedge \gamma^-_F(\bar{x}, a, s))]$
- D_0 is a set of axioms describing the initial situation S_0.

The central idea of situation calculus is to define preconditions D_a for each event to happen, and define the impact of these events on the world fluent D_s, once they occur. The primitive operator in situation calculus is *do(action, situation)*, which links up actions and situations. Thus:

$$A \times S \rightarrow S$$

The simple do operator can be iteratively employed together with the various axioms, to undertake more sophisticated operations like progression and regression, which can be used for planning and projection. A fluent F is true in a situation resulting from applying event a, to situation s, ***only if,*** either event/action a caused it to be true (listed under $\gamma^+_F(\bar{x}, a, s)$, or it was already true in situation s, and action a, did not cancel it (listed under $\gamma^+_F(\bar{x}, a, s)$).

Now, based on this formal definition of situational calculus, event control of a cyber physical environment can be represented. Let's say system uncontrollable events i.e. events outside the system, undertaken at time instance k be defined as $U(k)$. These actions are captured by the sensors and represented as $A_{ex}(k)$ and is defined as $A_{ex}(k) = f_1(U(k))$. And let's say the system controllable events taken at time instance k be defined as $C(k)$. These actions are system actions and can be captured by analysinzg the system operation cycles and lets represent them as $A_{sys}(k)$ and is defined as $A_{sys}(k) = f_2(C(k))$. Now, the of the dynamic environment at cycle k, can be affected by either the user-driven actions

$A_{ex}(k)$ or by system's own control actions as computed during the previous cycle i.e. $A_{sys}(k\text{-}1)$. If a controller is designed to capture the situation based on these events then, the net input to be provided to the controller can be defined as: $Inp(k) = f_3(A_{ex}(k), A_{sys}(k\text{-}1)$.

Then the state of the system in cycle k, is defined as: $S(k) = f_4(Inp(k), S(k-1))$.

The control action or the output of the system in a situation based control system is dependent on the state/situation and the Goal G i.e. $A_{sys}(k) = f_5(S(k), G)$. Finally, the translation of the controller output actions from a decision level to their physical implementation is undertaken via the Actuators: $Y(k) = f_6(A_{out}(k))$.

This builds the basic formalism for modelling the different types of events in cyber and physical world and the relations between them. But, for temporal symbolic control systems, the functions f_4 and f_5 cannot be formulated as standard numerical functions, as we want to handle symbolic data. We need an explicit mechanism which allows us to define the state or situation S(k). These functions ca be detailed based on the semantics of RGOLOG (Reactive Golog) [238] which is a variant of Prolog and Golog that allows concurrent processing and reactivity which are not handled adequately in standard Golog [245].

For defining f_4, which relates mechanisms for translation from $S(k\text{-}1)$ to $S(k)$, based on input actions $Inp(k)$, we will consider the basic D_a and D_s (pre-condition and post-condition axioms), are well defined. Then this translation is quite intuitively handled by situation calculus through its primitive operation $D_o(A, S) \rightarrow S'$. Hence: $S(k) = Do(Inp(k), S(k-1))$.

Now we consider the semantics for f_5, and look at the definition of G i.e. the goal or the equilibrium state which the system is trying to achieve. Let us consider a set of condition-action rules with possible conditions which can move the system away from the equilibrium. The conditions listed become true in different situations($S(k)$) based on the inputs $Inp(k)$, acting upon $S(k-1)$. The input actions can be either external or system controlled. The control actions required to bring back the system to the equilibrium state are $A_{out}(k)$. In general, there can be up to n condition-action pairs which can be represented as: $\varphi_1(X_1) \rightarrow \alpha_1(X_1), \ldots \varphi_n(X_n) \rightarrow \alpha_n(X_n)$ where X_i contains all the free variables present in the definition. The goal state can hence be defined as one where none of the condition-action pairs gets violated i.e.

$$S_{Goal} \models (\neg\varphi_1(X_1) \vee \alpha_1(X_1)) \wedge \cdots \wedge (\neg\varphi_n(X_n) \vee \alpha_n(Xn)).$$

As can be noticed, $\alpha_i(X_i)$ contains multiple free variables and hence may require multiple steps or physical actions to attain the goal state. Further, the i^{th} control action, may potentially lead to the j^{th} violation condition, and hence multiple iterations of control action may be required.

The problem of finding the appropriate control action can be defined as the planning operation of Situation Calculus. Such a process is handled in Situation Calculus literature based on proving a theorem [247] that:

$$\exists A_{out}(k) : Do(A_{out}(k), S(k)) \rightarrow S_{Goal},$$

where $A_{out}(k)$ is a sequence of control actions which are undertaken by the system one in each cycle. Note that this links the situation $S(k)$ and goal state S_{Goal} with the output action $A_{out}(k)$.

With these definitions, we can implement controller that can detect and act based on the situations derived from both the system internal and external actions. At the same time this model also allows one to keep track of chain of actions that have been undertaken to bring a non-equilibrium system into a stable system. These traces are the important steps for evolutionary actions which will be feed as Σ in definition 4.8 and eventually 4.10.

6.1.4. *SENSORIAL MODELLING FOR CPS*

Sensors and actuators form an important backbone of CPS. As defined in section 6.1.3, $Inp(k)$ and $A_{out}(k)$ i.e. inputs and outputs at time instance (k). But to support a complex ubiquitous computing environment, sensor networks will need to support localized cooperation of sensor nodes to perform complicated application directed tasks and in-network processing to transform raw data into high-level abstract information which is not necessarily a measurement the physical sensors themselves can provide. So, it's important to define formal abstraction for modelling the sensorial network of the CPS system. In a generic term sensors are devices that are capable of reading physical and physiological parameters.

Definition 6.12. A sensor S can be defined by a tuple $<D, C, C'>$ where

- D represents data-stream and is defined as $D=<DataType, \{R_1 \ldots R_n\}>$. $DataType$ defined the datatype of the data that is collected by the sensor and $\{R_1 \ldots R_n\}$ is an ordered set of readings obtained at various time s.t. for a time instance k $A_{ex}(k)=f(R_k)$, where function f is used to generate event from the sensor reading at time instance k.
- C the context of the sensor C and is used for understanding the contextual use of the sensor and can have a URI to an external Sensor Networks Ontology or can also be linked with other instances of metadata resources.
- C' defines the configuration of the sensor S, which includes parameters that define the working configurations of the sensor such as protocols, i/o data ranges, i/o interfaces etc.

So, based on this definition sensors that are part of the system can be modelled. After defining the sensors, we need to define actuators. Raw definition of an actuator is that it is a mechanism that puts something into automatic action. From electrical engineering point of view actuators are a subdivision of transducers and they are devices which transform an input signal (mainly an electrical signal) into some form of motion. Thus, a generic

purpose of actuator is action, thus can be related to $Y(k)$ as defined in definition 6.11. Now, let's formally define.

Definition 6.13. A actuator A can be defined by a tuple $<A, C, C'>$ where

- C and C' are exactly the same as defined in definition 4.12.
- A represent a sequence of actions i.e. is an ordered set represented as $\{A_1 \ldots A_n\}$ ordered set of actions to be undertaken by the actuator at various time s.t. for a time instance k, $A_k = f'(A_{out}(k))$, where function f' is used to generate action for actuator from the outptut action of the situational controller as defined in definition 6.11.

So, based on this definition sensors and actuators that are part of the system can be modelled and related to the situation based controller defined in section 6.1.3.1.

But to model a complex CPS, we need to introduce the concept of virtual sensor (VS). Virtual sensors originate from the works such as [248,249] and is used to provide feasible and economical alternatives to costly or impractical physical measurement instrument, by utilizing the functionalities of existing physical sensor. A virtual sensor uses information available from other measurements and process parameters to calculate an estimate of the quantity of interest. Abstractly, a virtual sensor is defined in terms of the physical devices that contribute to its construction. VS are termed virtual because they don't have physical occurrence but are formed by logical infusion of various sensors.

Definition 6.14. A virtual sensor VS can be defined by a tuple $<S, O, D, C, C'>$ where

- S is set of data sources such that $\underline{S} = \{S_1, \ldots, S_n\} \cup \{VS_1, \ldots, VS_m\}$ where S_i is physical sensor as described in definition 4.12 and VS_i is Virtual sensor that has been define before. Thus, existing VS can be used for creation of new VS.
- O is the operator relationship between sensors S. This operator is used to aggregate the data for VS.
- D is data-stream and is defined as $D=<DataType, \{R_1 \ldots R_n\}>$, exactly as in definition 4.12. The only difference is the reading R_n is computed by applying the operator O on the readings R of sensors S.

C *and* C' are defined exactly as the same of definition 6.12.

Thus, the most important part of VS is the operator O. The relation O can have different syntax and semantics as needed based on the type of data infusion that should be established between sensors. For instance if VS is used for taking into consideration humidity and temperature of a body part (e.g. arm, armpit) at the same time then a one to one union over the reading of the sensors for humidity and temperature will give an ordered set of reading for this new VS. And if the operator is to find the difference of the reading of two sensors (e.g. environment and body temperature) then the ordered difference between each element of the reading of two sensors will give the dataset for the

VS. Note that the context of the VS can always be obtained by the union of the individual contexts of the sensors forming the VS.

Another important aspect to be considered while providing complete sensorial model is ***Reinforcement Sensing (RS)*** which is important for distributed data sensing and to deal with sensor malfunctioning to still enable environment sensing at run-time. RS is based on the principle of utilizing data and information processing by taking advantage of data dependencies and redundancies that may exist in sensorial network due to components sensing the same or similar property (refer [250] and [251] for further understanding on reinforcement sensing also termed as collaborative sensing by other authors).

RS involves two computational steps:

(i) RS Analysis—identification of data dependencies and

(ii) RS Approximation—approximation of value for property *P*.

The RS Approximation is based on the dependency relations of P_i and thus can be calculated by using the similarity between the properties and associated concepts, which is calculated in the process of RS analysis. While, the RS analysis is a more complex task and bears the computational overhead of data collection vs. the readiness of dependency relation. In order to perform both the tasks the model of sensors and/or virtual sensors as given before play the most important role. Note that in the following section both the sensors and virtual sensors are denoted by *S.*

Definition 6.15. ***Reinforcement Sensing*** between two sensors S_x and S_y can be defined by the tuple $< S_x , S_y, \varphi, \mu >$ where φ *and* μ are the functions used to calculate the similarity vector between the two sensors, φ for calculation of semantic distance between the sensor context and μ for the data similarity.

Let us first provide the details on the methodology for calculating semantic distance i.e. φ. In this process for similarity measurement, weight allocation is made between the concepts nodes representing the sensor model, more precisely C and C' as stated in definition 6.12 are concepts from ontology that have weights allocated for all the relationships between concepts. For two concepts C_1 and C_2 which are related with the relation sub-class (*sub*), the weight allocation is calculated as:

$$w[sub(C_1, C_2)] = 1 + \frac{1}{k^{depth(C_2)}}$$

Where, *depth(C)* presents the depth of concept *C* from the root concept to node *C* in ontology hierarchy, *k* is a predefined factor larger than 1 indicating the rate at which the weight values decrease along the ontology hierarchy.

This formula has two important properties:

1. The semantic differences between upper level concepts are higher than those between lower level concepts, in other words, two general concepts are less similar than two specialized ones.

2. The distance between sibling concepts is greater than the distance between parent and child concepts. Specially, the depth of root concept is zero, and the depth of other concepts equal to their path length to root concept node.

In addition, if there exists multiple inheritance relation between concepts (such as, $C_1 \subset C_3, C_1 \subset C_2$), the depth of the concept node C_1 can have multiple values, and thus the weight will have multiple values. This process allows calculating the similarity between the sensors under consideration. If the distance is very high then the following step for data approximation can be ignored. An algorithm to achieve the implementation of this strategy is presented in Appendix A- section 11.1.

Now, in order to regenerated the data for sensor that failed which is semantically similar to some of the sensor that are working fine, let us assume that the sensor data manager collects the values of preselected data fields of the set of sensors in the latest time instances $t = 1. . n$. Furthermore, for acquiring the dependency relation, RS Analysis checks all the aggregated data to find out which data fields are dependent on others.

Let $S . d_j{}^t$ be the value of data field d_j of sensor S_i at a time instance t.

Hence $\{ S_i . d_j \}_1{}^n$ denote the time series of the data field d_j of component S_i at time instances *1 to n*. Now, let $\mu_{k_j}(S_x . d_j{}^t, S_y . d_j{}^t)$ be the distance between two data values of $d_j{}^t$ in components S_x and S_y measured by metric μ specific to d_j.

Then, for all component pairs S_x , S_y $x \neq y$, having the fields d_m and d_n, RS Analysis computes the boundary Δ_{d_j} such that the implication $\mu_{d_j}(S_x . d_j{}^t, S_y . d_j{}^t) < \Delta_{k_j} \Rightarrow \mu_{d_m}(S_x . d_m{}^t, S_y . d_m{}^t) < \Gamma_{d_m}$ for the time instances $t = 1. . n$ is satisfied in (at least) the specified percentage of all the cases (confidence level α_d, e.g. 90%).

Here Γ_{k_m} represents the tolerable distance threshold and is provided for each d_m. Based on this definition the RS Analysis concludes that the value of $S_x . d_j{}^t$ is close to the value of $S_y . d_j{}^t$ (and vice versa) for t such that the values of $S_x . d_l{}^t$ and $S_y . d_l{}^t$ are close as well. Thus, when a sensor S_i fails to sense the values of d_m, an approximation of this property has to take place. This is done by creating data stream with the exchange function $S_x . d_m := S_y . d_m$ and membership condition $\mu_{d_j}(S_x . d_j{}^t, S_y . d_j{}^t) < \Delta_{k_j}$.

If more than one S_ys satisfies the membership condition, an arbitrary one is selected. Thus, created data stream becomes the collected data for failed sensor.

Note that the task to compute the boundary Δ_{k_j} is resource and time demanding but some techniques that can be applied to lower the time such as sorting the data according to $\mu_{d_j}(S_x . d_j{}^t, S_y . d_j{}^t)$ or using sampling of the gathered data to obtain a statistically significant answer. Some other techniques such as linear regression, k-nearest neighbours, neural networks etc. can be applied to detect dependencies between data.

With the modelling of sensors, actuator, virtual sensors and reinforcement sensing model, we have formed a formal way of modelling the sensorial model of CPS. This model is

closely related to behavioural modelling of CPS (section 4.1.4), which eventually is related with the formal model of CPS (section 4.1.3) and evolutionary computation model (section 4.1.2). Thus, up to this point we have formulated a theoretical background on the top of which eCPS can be realized.

6.1.5. *KNOWLEDGE INJECTION IN EAS*

In general understanding knowledge represents the fact or condition of knowing something with familiarity gained through experience or association. In the context of computing system knowledge or more precisely knowledgebase is an organized repository consisting of concepts, data, objectives, requirements, rules, and specifications. If properly represented knowledgebase plays an important role for development of systems that are capable of reasoning and information retrieval by applying AI based techniques. Knowledge representation and reasoning (KRR) is the field of artificial intelligence (AI) dedicated to representing information about the world in a form that a computer system can utilize to solve complex tasks such as diagnosing a medical condition or having a dialog in a natural language. A knowledge base is an integral part of any knowledge-based intelligent system. It maps objects and relationships of the real world to computational objects and relationships. Knowledge representing a specific problem domain is often termed as domain knowledge. Domain experts define and design the domain knowledge and the process is commonly termed as Knowledge Engineering (KE). The generated knowledge can be represented in different formats, ontologies being a widely followed standard in the recent times. Ontologies provide formal naming and definition of the types, properties, and interrelationships of the entities that exist for a particular domain of discourse.

In the methodology for injecting knowledge in EAs, there are can be two tracks. The first way is the knowledge is extracted and injected in the EAs before starting the search for solutions i.e. the knowledge remains constant along the entire evolutionary process. While in the secondary approach, the knowledge is dynamic i.e. knowledge extraction and incorporation in the EAs continues during the evolutionary process. In this case, the knowledge is updated throughout evolution. Thus, these algorithms evolve a dynamic adaptation of the parameters, the evaluation or the genetic operators. In both the strategies the first and foremost stage will be the creating the links between the problem domain and the evolutionary algorithm. Figure 6-1 shows the overall methodology for incorporating domain knowledge into the evolutionary algorithms.

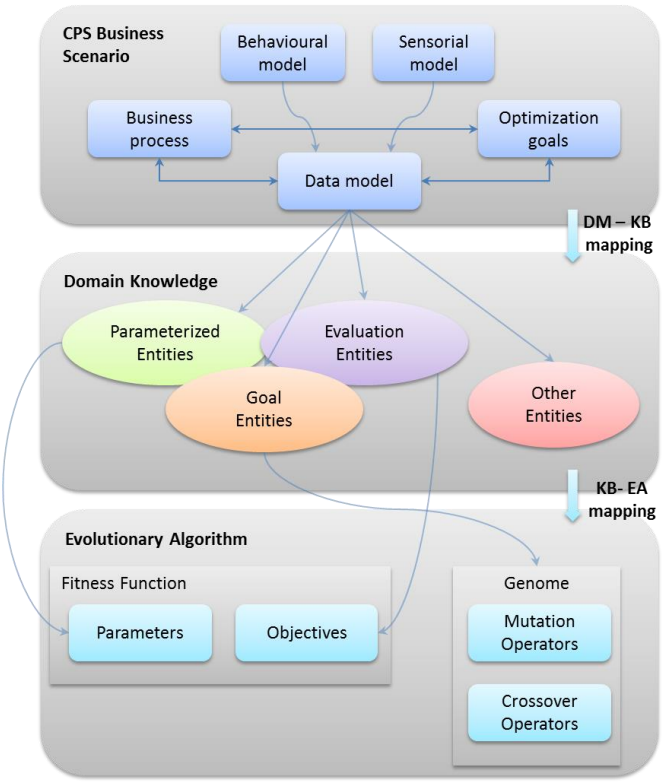


Figure 6-1 Methodology for injecting domain knowledge into EAs

The first step is to define the CPS business scenario. The central element in this process is Data model. The data model needs to deal with a huge quantity of information, taking into account all the constraints that can capture the overall business requirements. In defining the business scenario, the major entities are business processes and corresponding optimization goals. The sensorial and behavioural models (as defined in sections 6.1.4 and 6.1.3 respectively) are also liked up with the data model to define the overall system.

The next stage is creating links between the data model and the domain knowledge base i.e. DM-KB mapping. In this process ontologies are used- ontology is a knowledge representation of a domain or a field that provides conceptual resources for knowledge-based systems (KBS). It gathers and defines the set of objects that are known as belonging to the domain. In general, ontology is composed of entities sometimes called concepts or classes and relationships between these entities usually called roles, properties, or attributes if they are mono-valued. In fact, ontologies provide the conceptual and notional resources needed for knowledge formulation and for making knowledge explicit. The domain ontology formalizes the main concepts concerning the business domain. The domain ontology is composed of four general, overlapping categories of domain entities:

1. ***Entities to Optimize***: entities whose values are to be determined by the optimization algorithm (e.g., which ambulance will take care of each client). The

structure of the chromosome will largely depend on these entities and their relationships. It prefigures the output data structure of the evolutionary algorithm.

2. ***Parameter Entities***: entities whose values act as costs or constraints (e.g., cost/km for a vehicle, legal maximum number of daily driving for an ambulance attendant). The calculation of the fitness functions depends on these entities. They prefigure the structure of the input data.
3. ***Evaluation Entities***: entities whose values are objectives to optimize (e.g., minimize the total number of kilometres travelled by all ambulances or maximize the benefits). It must be ensured that each of these goals is represented by one or more fitness functions in the evolutionary algorithm.
4. ***Other Entities***: entities that are not involved in the optimization (e.g., history of customer calls to the customer call centre). The first three categories overlap. In fact, the membership of an entity to a category depends mainly on the usage of the values of its properties in the optimization problem. Some of the entities in the application domain may be “heterogeneous" when not all their properties serve the same purpose. In the frequent case where the function to be optimized must be integrated into a larger existing system, most entities already exist in its data model. Most often, however, new entities will have to be introduced to have a more fine-grained representation of the entities to optimize.

The first three categories overlap. In fact, the membership of an entity to a category depends mainly on the usage of the values of its properties in the optimization problem. Some of the entities in the application domain may be “heterogeneous" when not all their properties serve the same purpose. In the frequent case where the function to be optimized must be integrated into a larger existing system, most entities already exist in its data model. Most often, however, new entities will have to be introduced to have a more fine-grained representation of the entities to optimize.

In order to further detail this process of defining the domain ontology we define two high-level e generic entities i.e. *DomainEntity* and *EAEntity*. All the domain entities are defined as subclasses of *DomainEntity*. While, the entities specific to the EAs are defined as subclasses of *EAEntitiy*. Furthermore a generic object property *EAProperty* and three mutually exclusive sub-Properties *EAEvaluationProperty*, *EAOptimizableP roperty*, *EAParameterProperty*. All object properties of the domain ontology involved in the genetic algorithm must be subproperties of one of these three specific properties depending on whether they are involved for optimization, in the evaluation, or as a parameter. For the same purpose, but for atomic domain knowledge, we also define a generic data property *EADataProperty* and three mutually exclusive *subdataproperties*. The entity *EAEvaluationEntity* is defined as a subclass of *EAEntity* having at least one evaluation property such that:

- $EAEvaluationEntity \sqsubseteq EAEntity$,

- *EAEvaluationEntity* ≡ (*EAEvaluationProperty* some *Thing*) ⋁ (*EAEvaluationDataProperty* some (*boolean* ⋁ *dateTime* ⋁ *integer* ⋁ *real* ⋁ ...))

EAParameterEntity and *EAOptimizableEntity* are defined similarly.

The next step is to create the link between the domain ontology and the EA, which is termed as EB-EA mapping. This step is dedicated towards the definition of the fitness functions, the structure of the chromosome and the associated evolutionary operators which are the core properties of any EA. As depicted by the links in Figure 6-1, the parameters of the fitness function are linked with the ***Parameterized Entities***, while the objectives of the fitness function is linked with the ***Evaluation Entities***. Then, we need to similarly define the Genome in which, the mutation and crossover operators are linked with the ***Goal Entities.***

It can be observed that the use of domain entities in the evolutionary algorithm provide two levels of granularity in the ontology: properties are classified according to their use within the EA, and every entity involved in the realization of the EA will be automatically classified into one or more sub-classes of *EAEntity* based on its properties. Thus, it is obvious that there will some entities that will play the role of both as a parameter and as an entity to optimize. The use of domain knowledge in EAs will be further detailed with specific scenario in section 7.3.

6.1.6. *GENERIC MODEL OF ECPS*

In the previous sub-sections we have provided abstract models for different abstract machines composing the eCPS. To have a global view of the eCPS model it's necessary to clearly mark the links between the basic models. Figure 6-2 presents the abstract model of eCPS, which shows the basic abstract models with the parameters used in the definition and connection between them. Note that a typical abstract machine consists of a definition in terms of input, output, and the set of allowable operations used to turn the former into the latter. In the process abstract data types have been specified in terms of their operational semantics on an abstract machine.

The overall model is composed of four important abstract models i.e. Evolutionay Computation Model (ECM), CPS Model, Behavioural Model and Sensorial Model. At the top of the model is ECM, which provides the abstract model details for specification of evolutionary machines including eCPS. ECM is composed of EA, BEM and eCPS. EA provides the abstract definition of all forms of evolutionary algorithms. The populations definition in EA are related with the parameter *X[t]* i.e. the population at time *t* defined in the *Basic Evolutionary Machine (BEM).* While the automata *E[t]* is defined in relation with set of configurations S^H as defined in the eCPS model.

Furthermore the details of the eCPS are defined in reference to the behavioural and sensorial models.

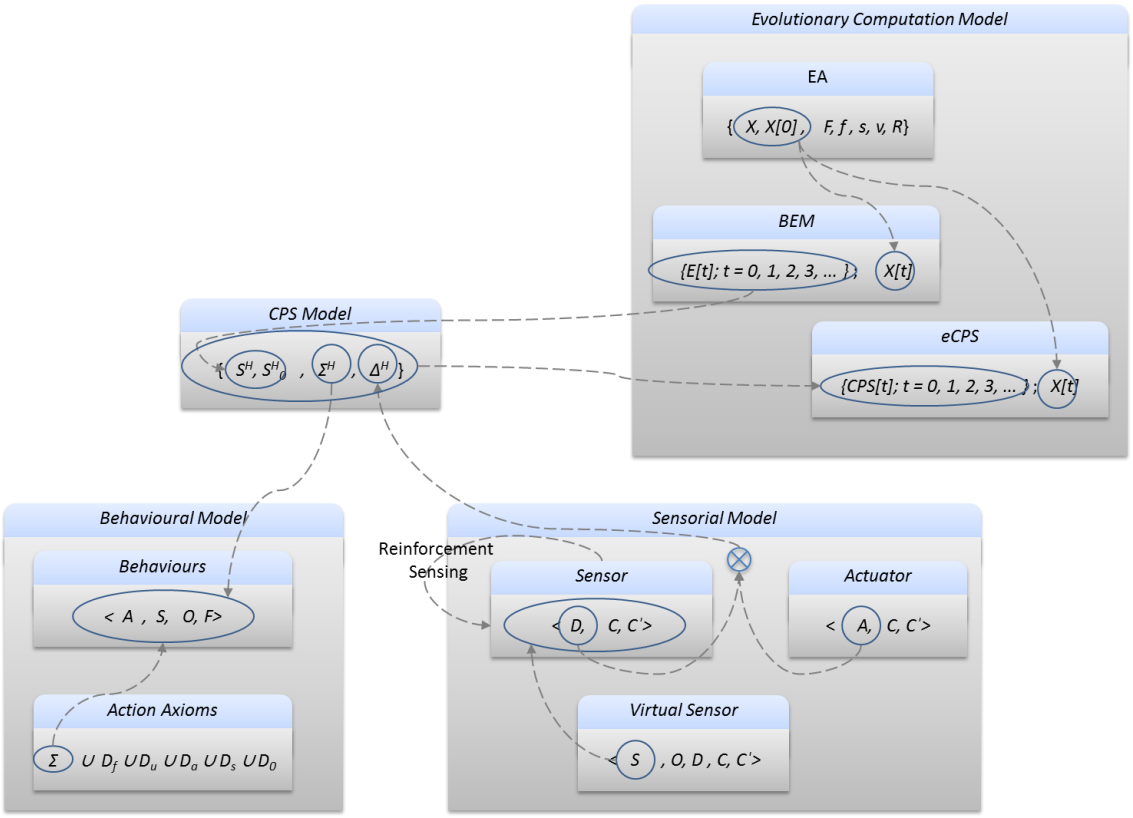


Figure 6-2 Abstract model of eCPS

The behavioural modelling is targeted towards defining configurations for the eCPS that are defining various states of the machine. While, the sensorial model, is used for defining the transitions that can occur from state to another based on the observations made in the environment. The transitions can lead towards the generation of actions that are enacted in the system via actuators. Thus, this model provides an abstract way of defining eCPS which has been implemented in different stages and is discussed in chapter 7.

6.2. Methodological and Technical Foundation

In the previous sections we have presented a formal way of representing evolutionary systems along with the formal modelling of system behaviour and sensorial system modelling fitting to the development of CPS and eventually eCPS. In this section and the following chapter we will explore CPS and eCPS from design and technical perspective. To keep everything intact and to see a global view, Figure 6-3 taken from [252], provides a self-explanatory view of evolutionary system model.

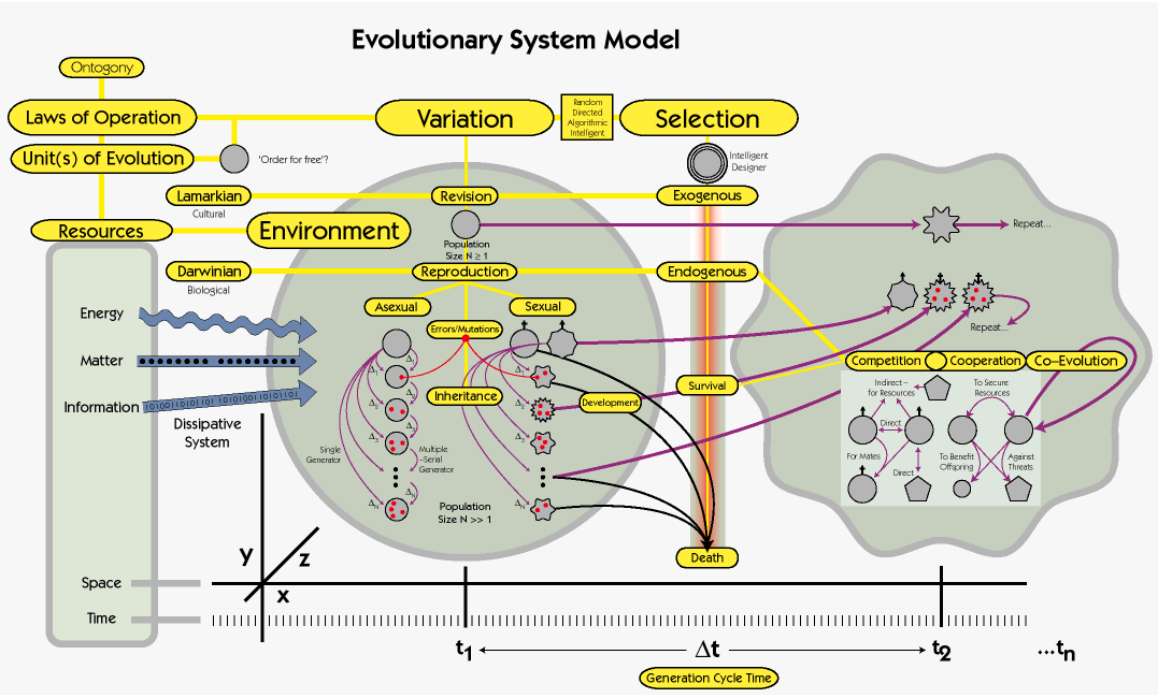


Figure 6-3 Evolutionary System Model

Most of the aspects in the above pictures have been previously discussed in section 3 and section 4.1. To recap the evolutionary process we know that evolution is affected by environmental factors and also affected by time and space dimension. Reproduction leads to variation which undergoes selection processes for adaptation to the changes. In the process of evolution competition, cooperation and co-evolution take place between the individuals fighting for survival and evolution.

In the following sub-sections, we will analyse evolutionary systems in a more methodological way. This will lead towards the formulation of the methodology for designing (e)CPS and lay foundation for technological foundation of (e)CPS.

6.2.1. *Reference Methodology*

CPSs are complex systems that have different components and stages during the entire system lifecycle. Technically CPS systems involve multiple layers and are often implemented with different technologies in each layer. This requires careful analysis of various stages of CPS evolutionary lifecycles and components necessary for implementation of different functionalities and associated technologies. The following subsections will provide the reference methodology for design and hence forth implementation of eCPS.

6.2.1.1. LIFE CYCLE OF SELF-EVOLUTIONARY SYSTEM

Evolutionary computation is based on the principle of monitoring, adaptation and evolution cycles. A number of monitoring and adaptation cycles have to be passed before a system can evolve, which is exactly the case in natural evolution. In this section, before we formulate the methodology for designing eCPS, we will discuss the overall lifecycle of eCPS. Figure 6-4 shows the overall processes in the lifecycle of the runtime environment of eCPS. The phases are basically divided into monitoring and evolution planes. Monitoring phase has a continuous collection of data from the physical world and also the changes in system requirements if any, that reflects the current state of the overall system. The eCPS reference architecture should provide the methodology and interface to collect the data, rules for data acquisition, validation rules etc.

In the next phase the system takes the data collected from the previous phase and performs analysis over it, based on the other inputs like system constraints, applicable rules, domain knowledge and available reasoning algorithms. The analysis engine should be able to decide the best reasoning algorithm to be used based on the contextual information collected from the previous phase.

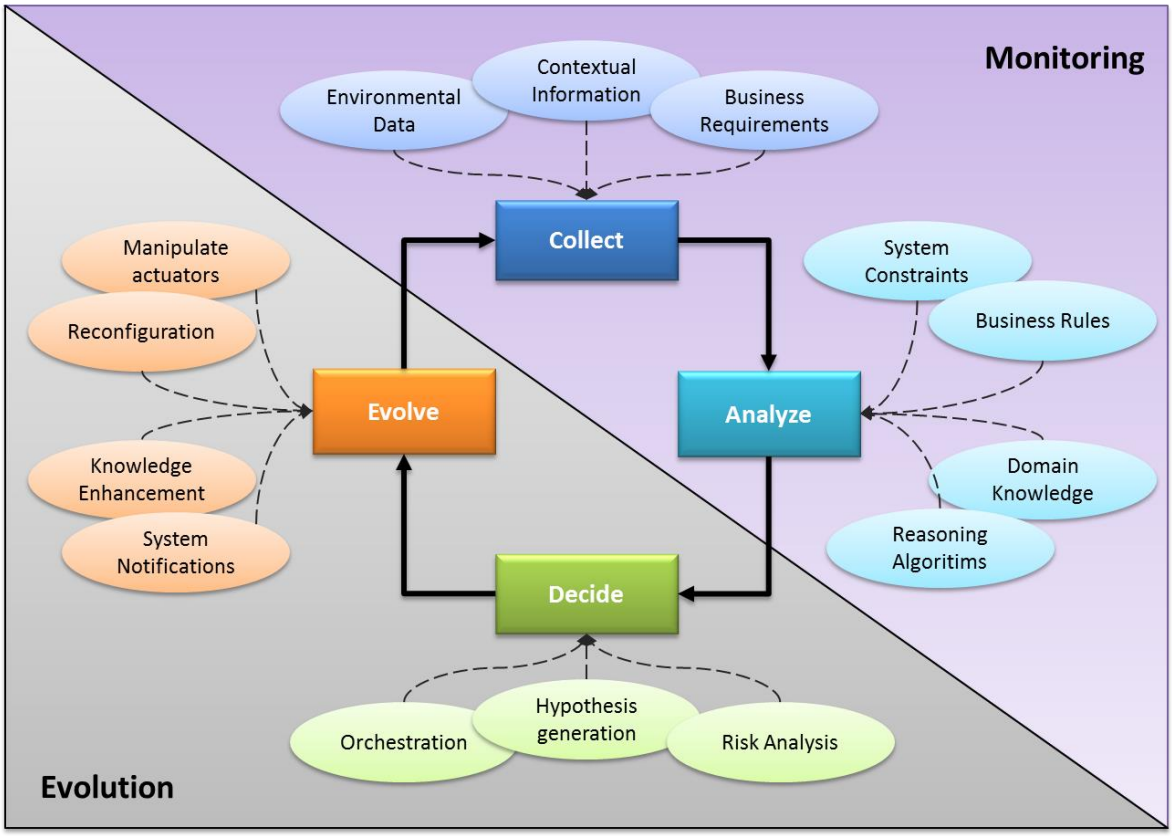


Figure 6-4 Overall life cycle of eCPS during runtime

Next, the system makes a decision and act accordingly about how to adapt or provide necessary evolutionary in order to fulfil the requirements from the analysed changes. This phase takes into account of methodology for risk analysis; hypothesis generation based on

reasoning over the domain knowledge and provides new orchestration planning with the available services or components. Finally, to decisions or actions formulated have to be implemented in the overall system without distorting the global performance of the system. At this point new system requirements blue prints will be generated for the run-time environment of the system, the acquired knowledge should be fed into the knowledge representation framework and if necessary adjust the actuators for interaction with the physical world. System should maintain proper logs of the changes, any non-strict rules violations and the evolutionary path taken by the system to authorized users.

6.2.1.2. DESIGN APPROACH FOR CPS

In this section we will provide a generic methodology for designing robust CPS and adapt that towards the development of eCPS. The design of a complex cyber physical system — especially one with heterogeneous subsystems distributed across networks — is a demanding task. Model-based design (MBD) [253–255] emphasizes mathematical modelling to design, analyse, verify, and validate dynamic systems. A complete model of CPS represents the coupling of its environment, physical processes, and embedded computations. At the same time modelled systems can be tested and simulated offline [256], enabling developers to verify the logic of their application, assumptions about its environment, and end-to-end (i.e. closed-loop) behaviour. In this section it is provided a set of steps, not necessarily sequential but necessarily co-dependent that facilitates the co-evolution of a model of CPS with its realization.

In order to model the cyber part of a system, designers need to implement the typical workflow for embedded systems design. This consists of modelling the functionality of a system in the functional model, and defining the architecture that can execute the functionality in the architectural model. Then, the tools provide a matching of the functionality onto the architecture to perform the functional-architecture co-design. While, to model the physical part of a CPS one needs to focus on the architectural model of the system first with mathematical models that describe the physical behaviour of the system. These models may involve differential equation solvers (e.g., the Euler method), when a discretized continuous time description is required.

Figure 6-5 depicts the generic workflow for designing CPS. The steps are discussed as below:

Step 1: Problem Definition – this step is composed of two important sub-steps viz. *State and Characterize Problem* and *Requirements Engineering* and will result in the generation requirements specification. In this step one needs to use simple language to describe the problem to be solved, without the use of mathematics or technical terminology. This is the “elevator speech” for the project and is a handy reference for developers, collaborators, colleagues and experts, vendors, and machine shops. In this step it’s important to take care of requirement tracking, metrics, formal testing processes,

and (most importantly) a process for peer review. Given the multidisciplinary nature of cyber physical systems, this step is necessary to effectively communicate design requirements. In this step it's also important to characterize the problem by isolating fixed parameters, adjustable parameters, and variables to be controlled. Identify quantities that characterize physical processes, such as configuration spaces, safety limitations, input and output sets, saturation points, and modal behaviour. Understand how a physical process may interact with a computation, including end-to-end latency requirements, fault conditions, and reactions to noise and quantization. Note that the environment on which CPS is going to be working provide the inputs for this phase of CPS design.

Step 2: Domain Definitions – this step takes the result of step 1 and makes clear distinction between the cyber and the physical parts of the problem to be tackled, which are depicted as steps 2.a. and 2.b. in Figure 6-4. The Architectural Model describes how the system is implemented, the hardware platform where the control algorithm executes, including embedded processors, sensors, actuators, communication primitives, operating systems, firmware, and drivers. While the functional model can be used to perform control validation, the architectural model contains values and formula used to model the physical quantities of interest, to simulate physical time, power consumption, associated costs, and other features of interest.

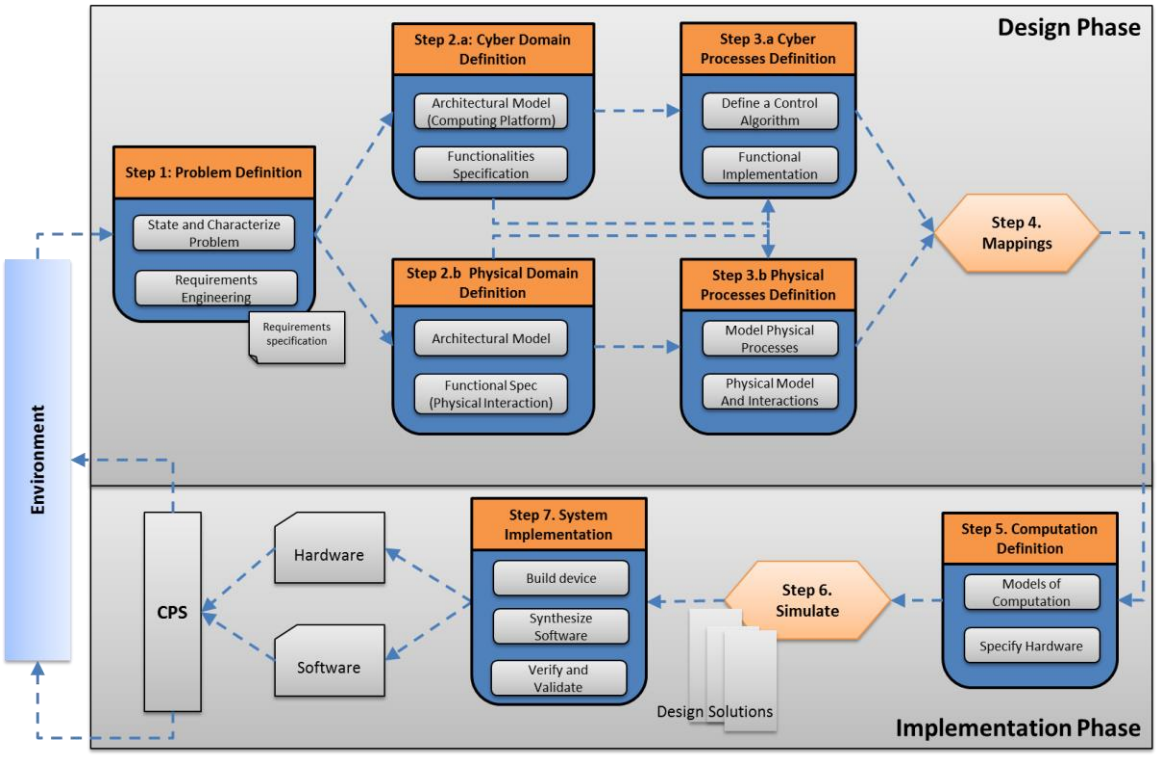


Figure 6-5 Proposed workflow to model CPS

Step 3: Process Definitions – this step is also performed independent for the cyber and the physical part of CPS and are depicted by steps 3.a. and 3.b. in Figure 6-4. In step 3.a., it is necessary to define and derive a control algorithm which determines conditions under

which physical processes are controllable and derive a suitable control algorithm to be executed by an embedded computer. Use the problem characterization to specify requirements on latencies, delays, sampling rates, jitter, and quantization so that the physical dynamics of interest can be accurately measured and suitably controlled; these requirements must be satisfied by the computational. While in Step 3.b., it's necessary to have a first iteration of physical modelling should establish basic observations and insight into relevant physical systems, such as the environment in which the cyber physical system resides, or physical processes to be controlled. Models of physical processes are simplified representations of real systems, and are usually in the form of systems of differential equations or Laplace transfer functions. What may had begun as simple mathematical models may need to be refined following development of a control algorithm, specification of hardware, and testing of components and sub-systems.

Step 4: Mappings – The proposed design approach is based on the separate definition of the functionality of the system, and of the possible implementation platforms. The two are then combined by mapping the functionality on cyber and physical world to generate the performance of interest in a structured way. s. In particular, this step allows architectural components to be synchronized with the functional components through mapping constraints, which schedule functions and their respective architectural implementations (tasks) simultaneously. This step thus provides a clear way of co-relating the cyber world model and physical world model and eventually provides effective guideline for computation model definition.

Step 5: Computation Definition – In this step its necessary to select the model of computation which is a set of allowable instructions used in a computation along with rules that govern the interaction, communication, and control flow of a set of computational components. A formal model of computation defines semantics that often result in greater analysability and the potential to simulate CPS through the use of heterogeneous modelling tools. Models described by formal models of computation may be easier to analyse with respect to determinism, execution time, state reachability, memory usage, and latency. These software dynamics alter the evolution of a CPS Note that due to the inherent complexity of many cyber physical systems often necessitates the composition of multiple models of computation. Advantages of using a specific model of computation depend on its semantics, whether timing constructs are used, and whether it is Turing-complete. Another important step in this phase is to specify and select hardware that is capable of withstanding the environment, interacting with the modelled physical systems, and implementing the control algorithm. For each component, consider its input and output bandwidths, delay from input to output, power usage, measurement resolutions and rates, and mechanical parameters such as form factor, rejection of electrical interference, durability, and lifespan. Mechanical actuators should be capable of producing forces and torques in excess of minimum values derived from earlier problem characterizations. Consider and model the impacts of using cost-effective substitutes for ideal parts; keep in mind that manufacturer specifications are not always accurate, and that hardware components should be independently tested. Selection of an embedded

computer may hinge on a deeper understanding of latency and execution time requirements of control algorithms, worst-case execution time measurements of synthesized software, and reasoning as to how software will interact with specific hardware architecture. This step may require several iterations with software design and simulation before an embedded computer can be selected with confidence.

Step 6: Simulate – In this step it's necessary to utilize some simulation tools like Modelica, Simulink, IBM Rational Rhapsody, Ptolemy II etc. If multiple models of computation are to be used, simulation and synthesis tools must allow the compositions of and interactions between multiple models of computation. Depending on the robustness of the development environment, incorporate models of sensors, actuators, and physical processes. Use platform-based design to separate application logic and architecture-specific software into modular components, which can improve code portability, reduce the impact of changing hardware components, and allow components to be reused in other contexts. Models of individual components and subsystems are as important as a complete end-to-end model. Component models provide a test harness for construction, verification of synthesized software, and testing. If no one modelling tool can completely describe the system, then for each subsystem use the modelling tool that best captures its dynamics. While disjoint simulations cannot represent relationships between signals that cross subsystem boundaries, or the behaviour of compositions of these subsystems, the exercise facilitates co-iteration of physical modelling, simulation, and testing. The output of this phase is a number of design solutions, among which the best fitting one has to be selected for implementation.

Step 7: System Implementation – This is a crucial step comprising of number of sub-steps. One important step is to build the hardware device according to specifications; taking note where exceptions have been made that may impact earlier modelling. Plan the development in a way that allows individual components and subsystems to be tested against theoretical models, which facilitates co-iteration between simulation and testing. The next step will be to utilize code synthesizer that directly support the embedded computer used, or generic code may be synthesized and tied to handwritten, architecture-specific code. Unlike many tools, models written in LabVIEW are natively executable across many platforms without knowledge of architecture-specific instruction sets or drivers, including desktop computers (for simulation or data acquisition), real-time processors, FPGAs, and ARM-based microcontrollers. In this step if code synthesis is infeasible or unavailable, customized code has to be written and should carefully follow the selected models of computation. And, the final step if verification and validation by using the test environment. It a good practice test each component and subsystem independently. Computational systems may be isolated from physical systems via hardware-in-the-loop testing, where programmable hardware such as embedded computers or FPGAs simulates the feedback from physical or other computational processes. Measurements of execution time and latency can be used to refine previous models, and unexpected test results may point to errors in modelling or implementation. Formal verification and validation give insight into the behaviour of an algorithm over all

or certain combinations of its inputs, or over the course of time. Precisely state requirements and translate them into a formal specification for verification and validation. Verification and validation are perhaps the most difficult aspects in the design of a CPS.

The results of these steps are hardware and software components that comprise the CPS system. The CPS when deployed in its working environment now works through closed loop interaction with environment. In the following subsection, we will introduce necessary steps in the generic CPS development workflow to enable self-evolutionary feature.

6.2.1.3. DESIGN APPROACH ENHANCEMENT FOR ECPS

eCPS mimic the process of organic evolution which the driving process for the emergence of complex and well adapted organic structures. At a simplified level, evolution can be seen as the result of the interplay between the creation of new genetic information and its evaluation and selection. A single individual of a population is affected by other individuals of the population (e.g., by food competition, predators, and mating), as well as by the environment (e.g., by food supply and climate). The better an individual performs under these conditions the greater its chance to live for a longer while and generate offspring, which in turn inherit the (disturbed) parental genetic information. These basic principles are to be considered in the CPS design and implementation process so that we can inject evolutionary behaviour in the CPS.

In this subsection we will enhance the generic design approach for CPS as discussed in section 6.2.1.2, to add necessary steps for defining eCPS. Figure 6-6 shows a new step named step 5_ev, which will be performed parallel with step 5 depicted in Figure 6-5. This new step is named Evolutionary Computation Definition.

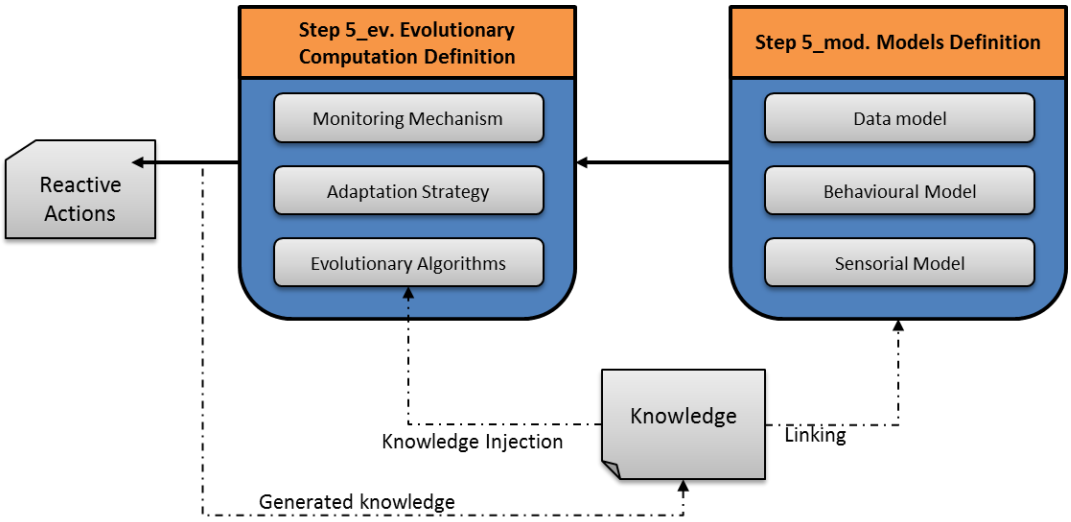


Figure 6-6 Enhancement of generic CPS design workflow for eCPS

Step 5_mod: Models Definition – This step is dedicated for specifying the models that will be used by the evolutionary computations processes of the eCPS. This step includes threes sub-steps i.e. specifying data model, behavioural model and sensorial model. Data model is dedicated for defining the data that related to the business processes that will be addressed by CPS and applied for evolutionary computation. Behavioural modelling is the step for specifying the behavioural model of the system as defined in section 7.1.3, which provides explicit mechanisms to model system changes and corresponding responses. This models defined in this sub-step is utilized for event detection, identification and feedback control loops which eventually will generate different action steps for the system to take under varying conditions. Behavioural model is dependent on the sensorial model which is another sub-step of this step. Sensorial model provides the specification on how the system will interact with the environment either to continuously collect the data or to actuate over physical systems working in the physical pane, which are based on the actions generated from the behavioural model. In the process of defining these various models knowledge which captures both domain or business specific concepts and capabilities is utilized with the mechanism of linking as followed in the process of creating linked data.

Step 5_ev: Evolutionary Computation Definition – At this step all the necessary components for self-evolutionary system is defined and is supported by the cyber and physical modelling in previous steps. At the same time at this step it's necessary to provide domain specific knowledge (formal knowledge) that describes the environment and the problem in a formalized way or machine interpretable way. When defining evolutionary computation model, the first step is defining monitoring mechanism, which is used for keeping track of the changes in the environment and/or requirements with time. This sub-step builds monitoring strategy based on environmental data, contextual information, system requirements and other available resources. The surges detected in the monitoring mechanism trigger the adaptation step. The second sub-step is adaptation strategy which defines how the system will or will not react to the changes detected in previous step. Adaptation mechanism can be based on rules based engine where rules are defined for expected environmental changes. But for uncertain cases hypothesis generation algorithms, automated risk analysis, reasoning algorithms etc. can be used. The adaptation mechanism definition should also include the mechanism to maintain traceability of the adaptation process, used algorithm and utilized domain knowledge. This generated knowledge map forms the base for evolutionary actions. Then the last sub-step is the definition of evolutionary algorithm(s). The results from previous steps for instance physical process modelling can be used for defining fitness functions of the evolutionary algorithms. Another important task in this step is injection of knowledge into EAs as defined in section 7.1.5.

Some important consideration is that over the course of evolution, this leads to a penetration of the system model with new information of individuals components which often help the performance to improve above-average fitness. This requires efficient mechanism to check the system equilibrium after the application of evolutionary steps. At

the same time the non-deterministic nature of variation leads to a permanent production of novel information and therefore to the creation of differing model of components, which need to seamlessly integrate into the system without having any hindrance in the performance.

6.2.2. *Reference Technology*

6.2.2.1. Reference Architecture

In presenting technological foundation, reference architecture is a well adopted methodology, which provides a template solution for architecture for a particular domain. Thus, formulation of reference architecture for eCPS is necessary to have commonality over components and functionalities. In the following sub-chapters, the technical solutions for the realization of eCPS are provided in details, which have also been followed in development of technical solutions discussed in the following chapters.

In this section is presented the generic reference architecture for CPS. Before presenting the reference architecture, let's discuss why the existing architecture of embedded system and real-time systems is not enough for CPS. [257] and [258] provide and interesting discussion embedded and real-time system reference architectures. In these architecture, sensor and actuator units are tightly coupled with a higher level control unit, and timing properties are carefully measured at each system level so that control loops perform correctly (both functionally and temporally). But, CPSs are a next-generation network connected collection of loosely coupled distributed cyber systems and physical systems monitored/controlled by user defined semantic laws. Based on these arguments a high level architecture of a CPS is shown in Figure 6-7. The main highlights of this architecture include:

- *Global Reference Time*: is provided by the next generation network. This should be accepted by all system components, including humans, physical devices, and cyber logic in this architecture.
- *Event/Information Driven*: just as human society is Event/Information Driven, future CPSs should also use a similar communication mechanism. Moreover, we differentiate between Events and Information. Events are either "raw facts" reported by sensor units/humans (called Sensor Events) or "actions" made by actuator units/humans (called Actuator Events). Information is the abstraction of the physical world made either by CPS control units or humans through event processing.

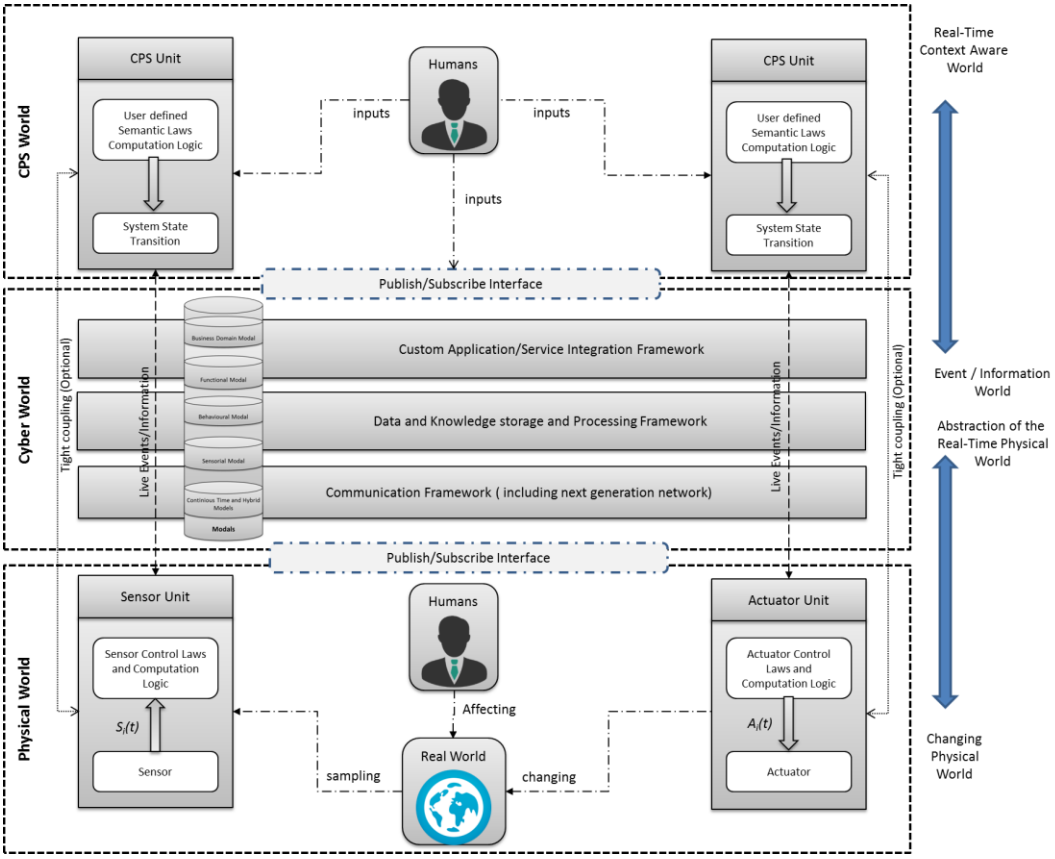


Figure 6-7 High level architecture for CPS

- *Quantified Confidence*: this design goal is archived by a Unified Event/Information model. Any event/information in this architecture should contain the following built-in properties. Global reference time – records the event/information occurrence/detected time. Life-span – specifies how long until that event/information's confidence level drops to zero. Confidence and confidence fading equation – specifies the event/information confidence level and how it fades over time. The confidence level and equation are decided by a particular device and control logic to provide a standard method for the subscriber to calculate the confidence of the subscribed event/information at any point in time. Digital signature and authentication code – specifies who published and who can access the event/information. Trustworthiness – specifies how much the subscriber trusts a particular publisher. Dependability – specifies the subscriber's dependence on event/information provided by a publisher in order to produce a particular outcome/information. Criticalness – specifies the critical urgency of each event/information so the subscriber can allocate its system resources based on different system design goals.

Now let's detail each of the components of the three important layers of the cyber world.

Communication framework

The framework is responsible to layer provides the components that facilities the communicating from the data sources across various communication protocol. One generic instance of this framework is shown in Figure 6-8.

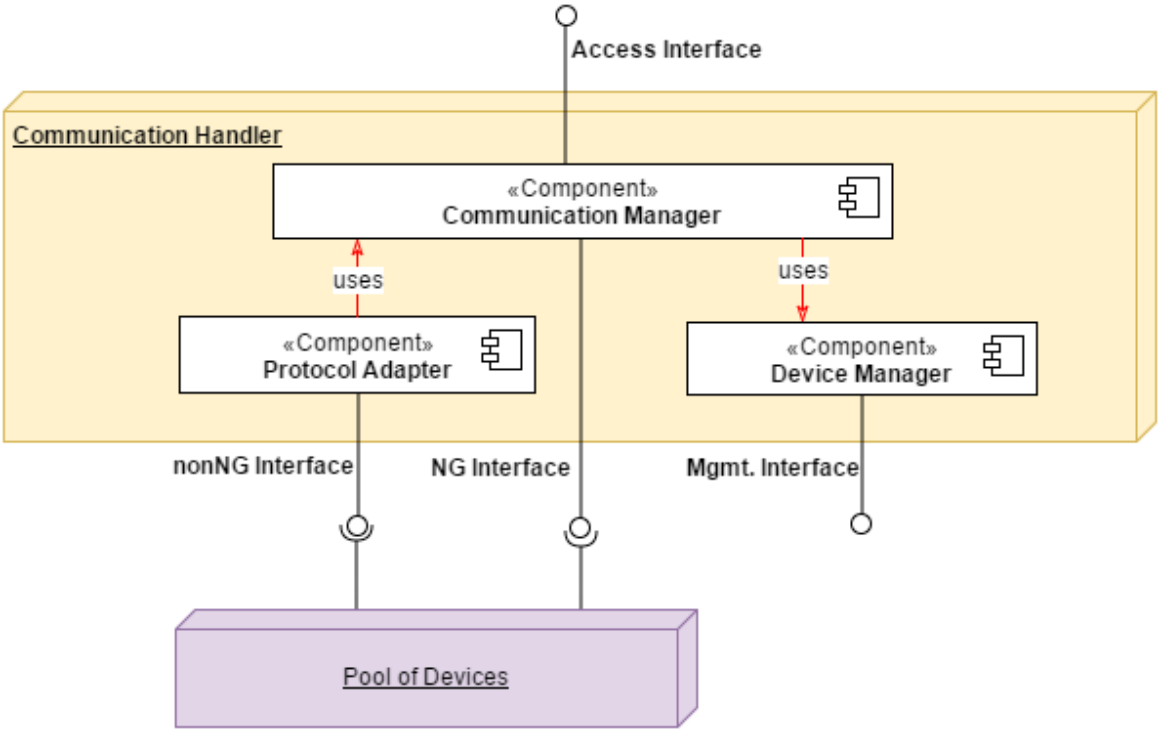


Figure 6-8 Communication framework instantiation

The top most component is communication infrastructure and it's considered here the next-generation network (NGN) as communication infrastructure because of the need of Global Reference Time for CPS. Refer [259] for detailed understanding of NGN. In reference to NGN OMA has also produced OMA Next Generation Services Interface [5]. But, not all the devices from the sensorial pool are NGN compliant. Thus, the component protocol adapter is introduced that acts as the adapter to enable NGN non-compliant devices to utilize the NGN communication infrastructure. For IoT paradigm supported standards like M2M are directly integrated through pub/sub client for device registration and continuous data collection. Other standards are supported by implementing protocol adapters. This layer thus is responsible for supporting various protocols like M2M, GSN, GS1 EPC ALE Service, ZigBee etc. and provides uniform protocol conversion for communication with higher layers. At the same the considered instance of communication framework also has the component device management, which provides the registration

[5] http://technical.openmobilealliance.org/Technical/technical-information/release-program/current-releases/ngsi-v1-0

and discovery functionality for all the devices connected the overall system. This is an important added value to devise more business and industrial oriented CPS such as in intelligent manufacturing, collaborative production etc.

Data and Knowledge storage and Processing Framework

This framework is responsible for processing the data collected from the lower layer and create useful knowledge and/or detect events. One generic instance of this framework is shown in Figure 6-9.

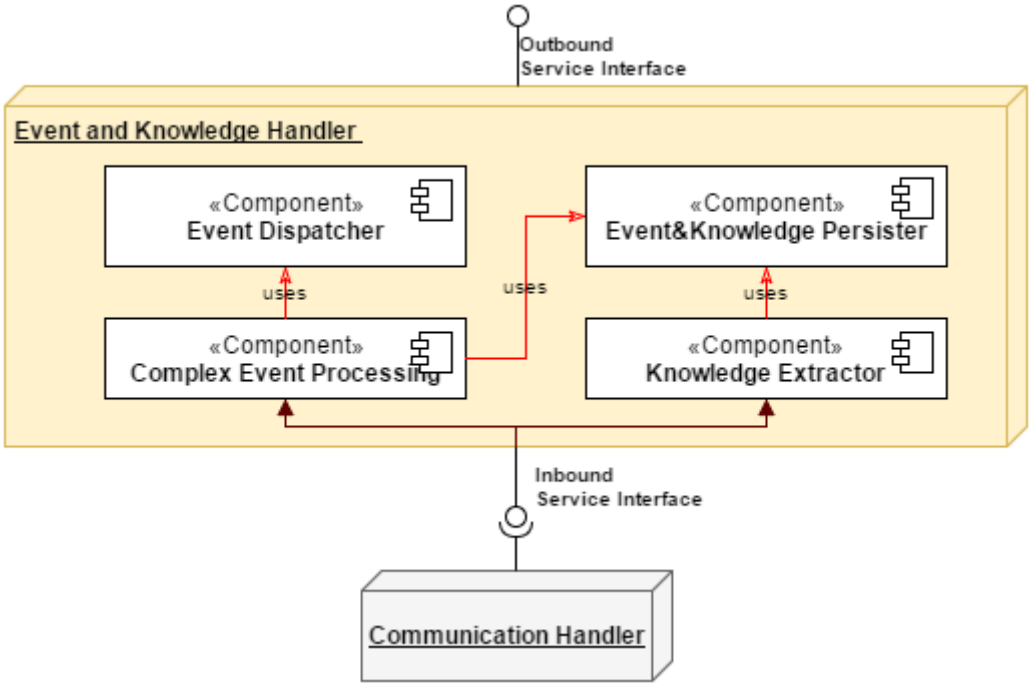


Figure 6-9 Data and Knowledge storage and Processing Framework instantiation

The lower-most component is the data filtering and aggregation components, which receives data from the communication layer. The data thus collected is filtered to remove faulty data based on some predefined rules. An example of filtering is range based filter i.e. remove all data that's outside the range for the given type of device. Similar action can be used to filter out incorrect data type based on sensor type. This action cleans the data and can avoid unwanted exceptions in higher levels. At the same time this component handles the instantiation of virtual sensors by utilizing the operator of VS, which is usually aggregation operator. Refer section 4.1.5 for understanding of VS. Thus produced data is passed towards Complex Event Processing (CEP) component and Knowledge Extraction (KE) component. CEP is event detection methodology by combining data from multiple sources to infer events or patterns that suggest more complicated circumstances. The goal of complex event processing is to identify meaningful events (such as opportunities or threats) and respond to them as quickly as possible. Refer [260] for a good understanding of CEP. The next component is KE which implements functionality for creation of knowledge from data streams coming from lower layers. The resulting knowledge needs to be in a machine-readable and machine-

interpretable format and must represent knowledge in a manner that facilitates inferencing. Thus generated knowledge and events are pushed towards the storage and retrieval component that provides interface for storing and retrieving events and knowledge along with their temporal semantics. While the events are also pushed towards event dispatcher component which immediately provides the event details to all the subscribers via the publish/subscribe interface.

Custom Integration Framework

This framework is responsible for providing easy and seamless integration of custom implementations into the core system. One generic instance of this framework is shown in Figure 6-10.

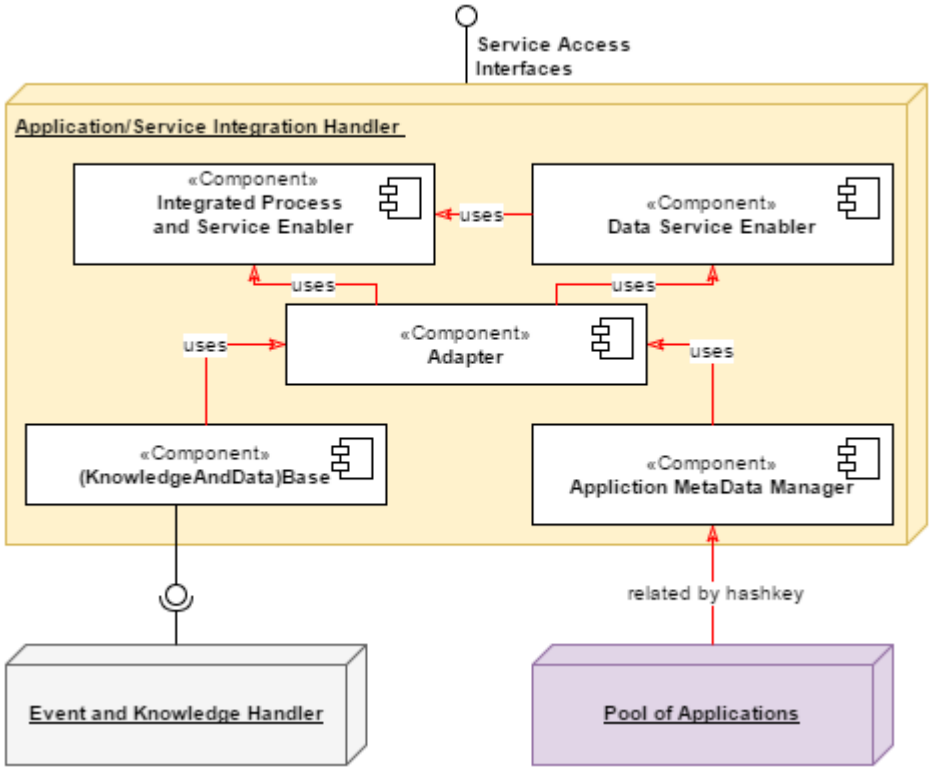


Figure 6-10 Application/Service Integration Framework instantiation

Adapters are the core mediator for integration process, which provide adaption over data and application logic. Database and custom application communicate with adapters for data and functional adaptations respectively. While the adapters communicate with the Process integration and service enablement and data service enablement components to provide service interfaces to access the data and application logic of custom application. This generic instance is based on the principle of Service Oriented Integration approach and follows the idea that the custom application is translated as service interface via adapter and thus created service interfaces are accessed by the service bus for integration. This reduces integration burden by simplifying service interactions. Follow [261] for further understating of SOA and application integration strategies. Some of the interesting

case for necessity of custom application/service will be discussed in chapter 7 where we will demonstrate some instances of the generic architecture.

The different components at each of the layers of the CPS reference architecture form the base of eCPS too. The additional functional implementation for realization of eCPS from CPS based system is the implementation of additional components for evolutional computation. The details of the component are as shown in Figure 6-11 and the way the different sub-components (population initializer, evolutionary operators, evaluation functions etc.) are explained in chapter 7.

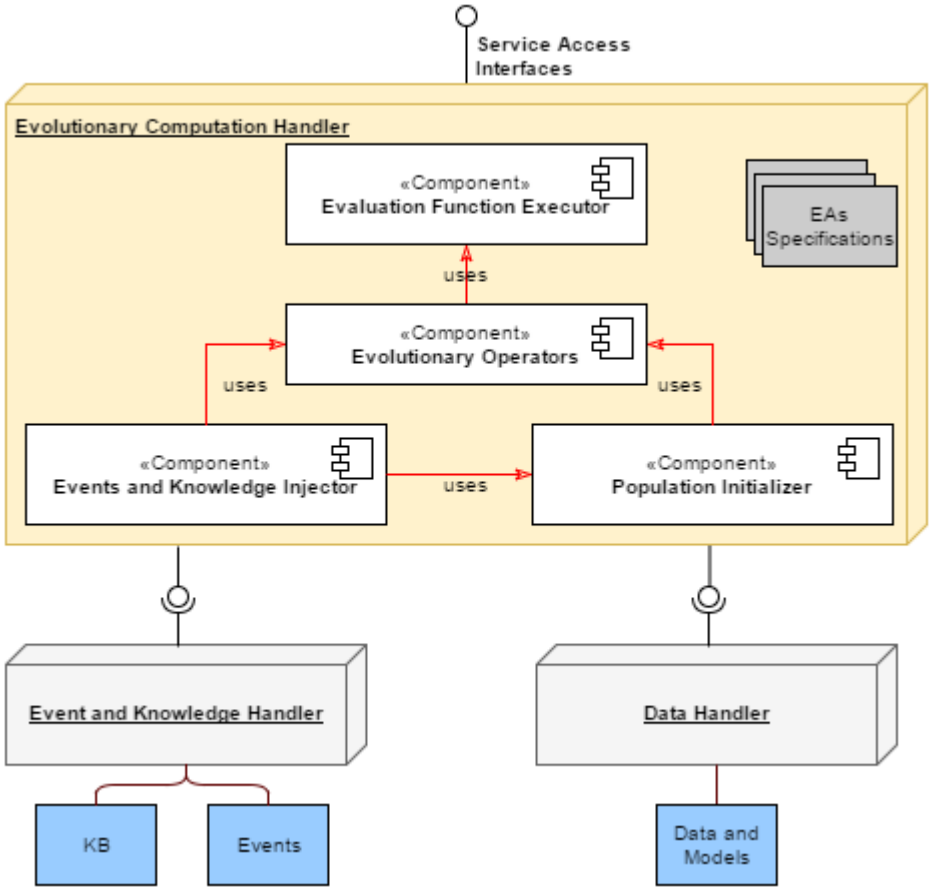


Figure 6-11 Evolutionary Computation Handler instantiation

This additional component is part of the custom applications layers in the generic architecture and service integration framework is utilized to integrate this custom component into the overall system. Note that this additional is necessary because eCPS is a special case of CPS with functional implementation for evolutionary computation. Rest of the architecture with data and control flow remains the same for both CPS and eCPS.

6.2.2.2. TECHNOLOGY BLOCKS

In this section let's try to understand the technologies that can be used to realize all the layers of the CPS as proposed in section 6.2.2.1. Figure 6-12 shows some of the important

existing technologies at each of the layer. The technology stack has more layers just to make distinction between some layers, but doesn't conflict with the generic architecture.

In the lowest position of the technology stack is the technologies related to the devices protocols which are used by different types of sensors and actuators. Among them IP/USB are often used by low level devices with less complexity while somewhat evolved devices use protocols such as ZigBee, ZWave etc. which are wireless protocols and thus can also be used as communication protocols in similar but not exact paradigm like IoT paradigm. The devices can be easily embedded in low level device hardware platforms such as RaspberryPI, Arduino etc. The benefit of doing this the added flexibility that we can achieve to implement and deploy protocol adapters to convert device level protocols to standard CPS communication protocol i.e. NGN. Above the communication protocols we have communication API and is composed of very recent interfaces such as Next Generation Service Interface (NGSI), Remote Procedure Call (RPC), XEP-0060 – the XMPP protocol extension for generic publish-subscribe functionality etc. These standards are very useful to define uniform publish/subscribe interfaces.

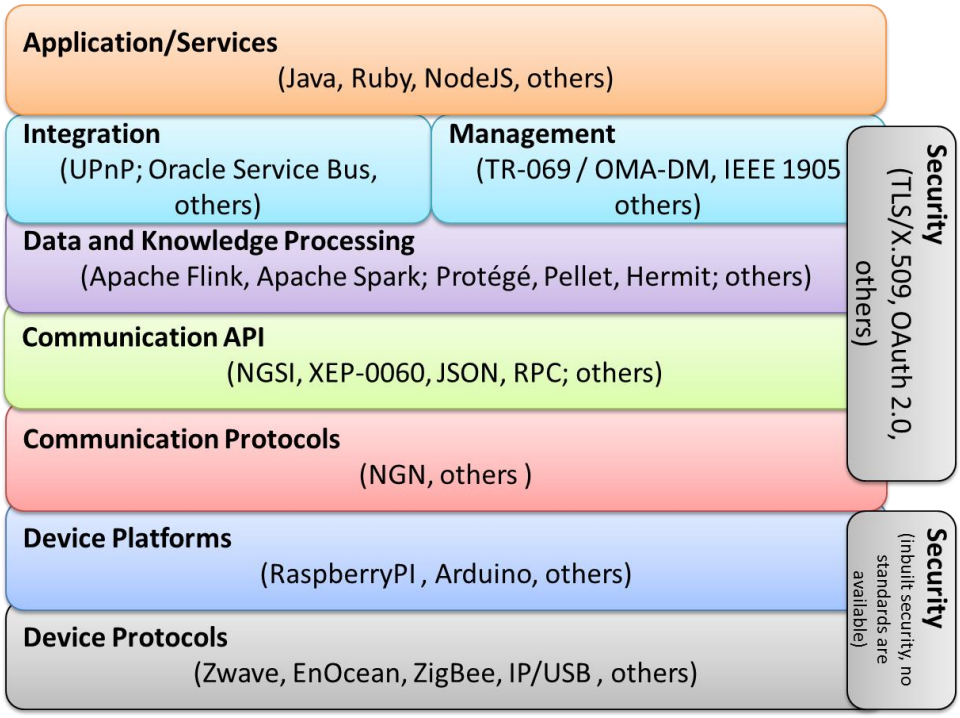


Figure 6-12 Technology Stack for CPS

From this point on, the layers get into high level functional necessities such as for data and knowledge processing. This layer is quite enriched with technologies and some prominent ones have been specified in the figure. But, also note that the use of these technologies requires proper configuration and adaptation based on the functional requirements of the CPS under development. The integration layer has not been explored much technically, but the recent protocol like Universal Plug and Play (UPnP) permits networked devices including mobile devices to seamlessly discover each other's presence

on the network and establish functional network services for data sharing, communications, and entertainment. But so far UPnP has not been explored in enterprise-class devices. And the Service Bus often termed as Enterprise Service Bus (ESB) such as Oracle's service bus provide solution for easy and seamless integration of applications and services. Management layer though not depicted in the architecture is an important layer when working with industrial CPS. In this layer there are some well accepted standards such as TR-069 / OMA-DM, IEEE 1905. For instance TR-069 ACS Management protocol provides specification for remote management of devices such as modem, routers, IPTV/ STBs, ATA/VoIP, storage devices, media centres, femtocell, IP-phones, cameras, etc.

The vertical layer is composed of security. For the lower layers, the security is dependent on custom implementation such as identity based authorization. While the upper layers can adopt some standard security protocols such as TLS/X.509, OAuth 2.0 which have been well accepted in almost all enterprise and/or industrial scenarios. The topmost layer i.e. the application/services layer is the most free layer and has lots of technologies to choose from and we have mentioned only some of them just based on our preference of usage.

6.2.2.3. DYNAMIC DEPLOYMENT FRAMEWORK

In the context of eCPS, it's highly necessary to address dynamic maintenance, adaptation and extensibility of software systems. In, this section the main objective is to provide a framework that adds up the capability to incorporate the changes on the overall system dynamics at runtime. To make it further clear, let's say the functional requirement of the system changes due to the new paradigm detected after *n* generations of EA execution. Then in this case the new functional requirements need to be implemented and injected back into the system. In general this a very complex task, just imagine replacing a car tyre when the car is running. Furthermore consideration of the system consistency and security guarantee increase the complexity of the dynamic deployment framework. In, the traditional sense, it's the automation of the task of the system administrator responsible to manage and evolve large scripts that control the system. The dynamic reconfiguration process looks very much like the traditional control system model of "sense-plan-act"..

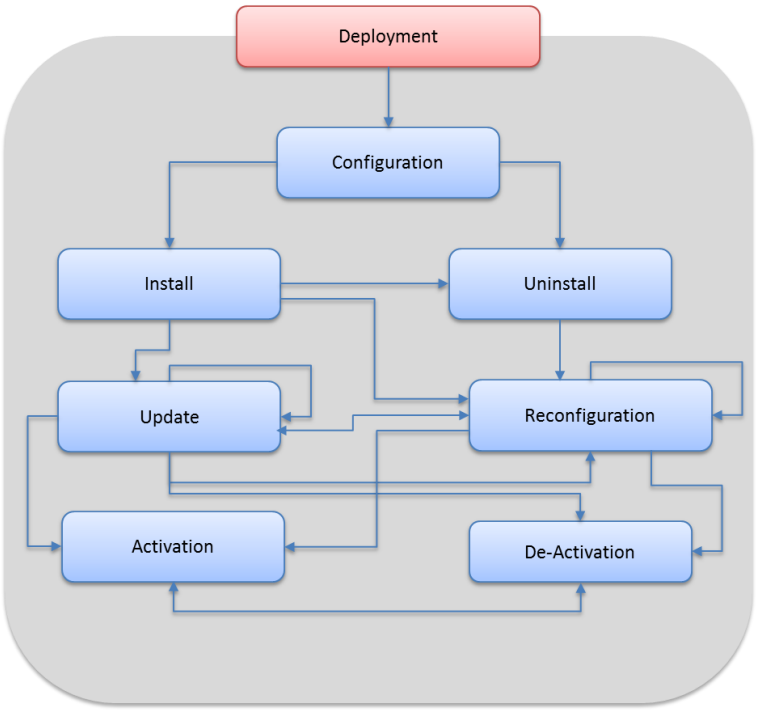


Figure 6-13 Software deployment life-cycle

For software systems, sensing involves the monitoring of the system and its environment to detect problems such as machine or component failures. Planning involves the construction of a plan to return the software system to normal or near normal functionality. Acting is the execution of the steps as defined in the plan. Each step in the plan effects a state transition. The plan as a whole causes a transition from the present state of the system to a desired state. The problem is that there are number of ways in which this state transition can be performed. All of these ways have different time, cost, and resource usage implications. Finding the optimal plan is difficult when all of these variables are taken into account

But, before going into the details of the dynamic deployment scenario, it's necessary to understand the overall software system deployment life-cycle. The main activities of the deployment life cycle are as depicted in Figure 6-13. The activities are:

- ***Configuration***: a component may have many versions and many implementations for different operating systems for example. The configuration consists of selecting the appropriate components according to the target platform, packaging them with their depending resources. According to the component model, a package may contain one or many components.
- ***Install***: consists of transferring the applications packages from a storage repository to the target site or sites.
- ***Activation***: after the components are injected within the target environment, they have to be loaded and started in order to use their services.

- ***De-activation***: for different reasons, a component or a set of components have to stopped and unloaded.
- ***Update***: consists of reinstalling some components already installed, or adding new components to an application previously installed.
- ***Reconfiguration***: means the modification of the installed application using the elements already available on the site (without need for re/installing other components). For example disconnecting two components or plugging a component in the place of another, both present on the site.
- ***Uninstall***: consists of removing from the target site the components no longer used.

Furthermore, in order to achieve dynamic deployment functionality, two important factors to be considered:

- It is necessary that information related to deployment have to be specified outside the software components implementation i.e. it's necessary to have metadata stored in some descriptor file for all the independent sub-components of the software system. The generic way to prepare the deployment descriptor is a set of key/value pairs.
- The other important point is related to the techniques used to modify the applications at run-time. The main idea for this technique is based on the ability to reify the connections between the application elements and to act dynamically on these connections. The reification can intervene at different levels i.e. virtual machine while the second approach is reification at the application level.

Among the two techniques for modification of the run-time deployment, it's necessary to have the mixed approach. The virtual machine based approach is useful when the overall system functional requirement undergoes changes. In this case the reification in VM provides the advantage that all applications running on top of the modified virtual machine benefit automatically from the dynamic adaptation capabilities. While the second technique is applicable when the application level changes are incorporated independent from other components. In this case the container can be a dynamic proxy (EJB/JBoss), a static generated interceptor (EJB/Jonas) or any other interception object. The interceptors are particularly inserted in order to apply some necessary system services (security, transactions, and mapping with databases) before transferring the calls to the wrapped components.

Based on these principles, here we will present the details for the dynamic deployment framework, the high level view of which is as depicted in Figure 6-14. The major components of the framework are: collection of software components base, collection of deployment and reconfiguration policies and there software components viz. deployment planner, deployment and reconfiguration manager and abstract models manager. Each of the components is described below.

Software Components Base: This is the collection of all the software components that are required for the functioning of the overall system. Thus, basically this is the collection of jars, wars, exes, code base etc. which needs to be deployed and executed at run-time.

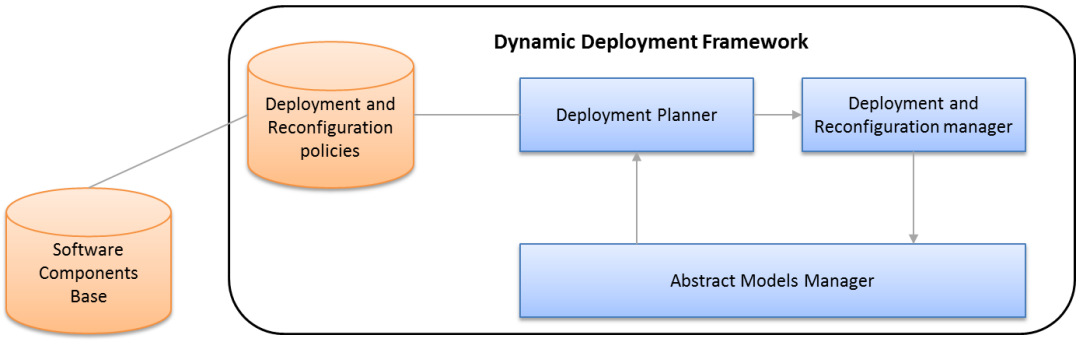


Figure 6-14 Dynamic Deployment Framework

Deployment and Reconfiguration Policies: This is the collection of policies that govern the configuration and deployment of the software components base. For all the components added in the software components base its necessary to have corresponding policy description added in this collection. Furthermore, it's necessary to define rules and policies to detect and trigger the deployment and reconfiguration when any of the software components undergoes changes. Some policies are required to make possible the self-reconfiguration of applications. Self-reconfiguration means the ability to take consistent dynamic reconfiguration decisions without the intervention of an external actor (usually a human administrator). The current specification of reconfiguration policies is very primitive. It considers a very simple form of reconfiguration rules. The improvement of this specification is one of our perspectives

Deployment Planner: This is the core component and the role of this component is to allow the auto-reconfiguration/deployment of applications. It represents a reasoning engine that introspects or receives events from the system and its environment and takes decisions according to these events and according to some reconfiguration and deployment policies. This component works is two phases viz. *Deployment Planning* and *Replanning*. *Deployment Planning* deals with the initial deployment the process by which the system is deployed across the network for the very first time. Initial deployment takes a domain, an initial state, and a goal state as its inputs, which are determined by the overall system goal and available software components, which are accessible via the deployment and reconfiguration policies base. The goal state specifies the normal operating state of the system in which all machines are up, all components and connectors are assigned to machines, and all components and connectors are connected. *Replanning* activity is performed when the system undergoes changes once the system is deployed and running. In the event of a change, the effect of those changes must be determined, the analysis of the effects of changes produces a new specification of the current state that reflects the fact that various components and connectors and virtual machines are affected. In order to make robust reconfiguration plan, it's important to incorporate some

intelligence in the deployment planner. The proposed methodology is to make use of conditional probability distributions derived by analysing historical attribute. Runtime setting for every component is hard to determine in advance due to the dynamic interaction of these components with the environment and the user. In this way the reconfiguration planner can take an advantage of probability theory and statistics to describe uncertain attributes. Knowledge about runtime uncertainty can be captured by a data structure for probabilistic inference called a Bayesian network (BN). A BN is a Directed Acyclic Graph (DAG) represented by a triplet (*N, E, P*), where *N* is the set of chance nodes, *E* is the set of arcs to represent causal influence of the chance nodes and *P* is the conditional probability distribution for each chance node. A Decision Network is a BN that also includes a set of *decision* and *utility* nodes. *Utility* nodes express the preferences among possible states of the world in terms of a subset of chance nodes and decision nodes. A probability weighted expected utility is calculated for each decision given the evidence. To represent variables that change over time, it is possible to use a time-sliced network such that each timeslice corresponds to a time point. A DDN is used for the enactment of the decisions that change over time influenced by dynamic states and preferences. To model the effectiveness of reconfiguration over time, decisions are modelled using a DDN where each time slice contains an action taken by the system. Refer [262] and [263]for more details understanding of DNNs

Abstract Model Manager: In defining a unified framework it's necessary to handle abstract models which are endowed by capabilities to support the dynamic deployment/adaptation. The abstract models allow the abstract definition of the components specifically defining their characteristics functions and service interfaces. With the help of these abstract models, the framework allows for example to replace a component by another one, either having the same interface or not (via the interface mapping facilities), it allows also to change the system architecture by changing the connections between components, by adding or removing components, it guarantees some consistency of the reconfigured system by assuring the state transfer between the old and the new component, passivating and activating components as necessary. By analogy with MDA the abstract component model corresponds to a PIM. This component provides the necessary details of the components to the deployment planner along with the possible reconfigurations suggestions based on the interfaces defined in the abstract models. While the deployment and reconfiguration manager provides necessary results of deployment to this component to check overall consistency of the system.

Deployment and Reconfiguration manager: It is the central part of the framework and provides and implements the basic dynamic deployment and reconfiguration routines. These routines operate on the system abstraction (and not directly on the base-level) which guarantees the independence and the genericity of the framework.

7. IMPLEMENTATION AND VALIDATION

And above all, watch with glittering eyes the whole world around you because the greatest secrets are always hidden in the most unlikely places. Those who don't believe in magic will never find it.

-Roald Dahl

This chapter presents the technological solutions that have been developed during the implementation phase of this research work. Some of the results that are presented in this chapter have been published in international conferences and research journals. The technological implementations cover different parts of the theoretical propositions made in the previous chapter. At the same time it's important to state that the technological solutions have been presented with different scenarios that have been tested with some industrial use-cases from various projects. The chapter presents three scenarios so as to divide the complete solutions into smaller use-cases solving problems in an agile fashion. By the end of this chapter the user is expected to have the insight into the implementation side of the evolutionary system along with understanding of the applications in industrial use-cases.

In this section several application scenarios are presented, those represent selected views of the framework that can be used for different purposes. All the technical solutions that have been developed in the scope of this research work follow the layered architecture as shown in Figure 7-1.

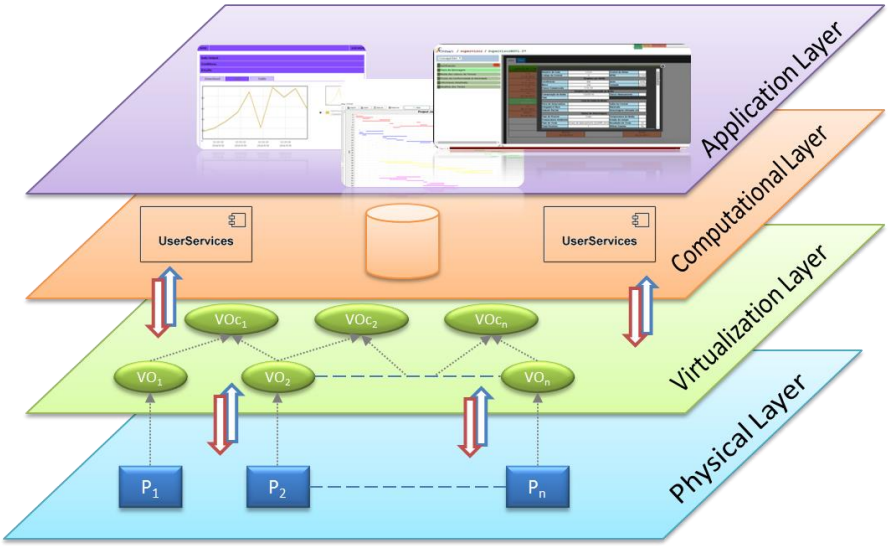


Figure 7-1 Layered architecture for CPS based applications

The lowest layer is composed of physical devices (P_x) which is pool of sensors and actuators, which have virtual instances termed as Virtual Object (VO_x) created in the virtualization layer. The virtualization layer can also create other complex virtual instances termed as Virtual Objects Complex (VOc_x) i.e. the ones without real physical existence by utilizing or more correctly logical recombination of the existing virtual instances. In the figure the dotted lines show the link between the physical devices and virtualized resource, while the coloured arrows depict the data/action flow. The

data/action flow is bidirectional in all layers. The virtualization layer is then followed by the computational layer which implements various user services for functionalities like data storage, data processing, events detection and other data analytics features. Finally the application layer is generated which provides the applications that are developed to meet different business scenarios including management and monitoring functionalities.

In the following sub-chapters three scenarios have been developed which are stacked together to achieve the evolutionary CPS application scenario. *Scenario 1* presents sensing framework, which provides the solution for integrating sensors and actuators into the system. This scenario is based on the theoretical foundation of sensorial modeling as presented in section 6.1.4. The solution formulated in this scenario is utilized on both the scenarios that follow. *Scenario 2* presents the solution of situational awareness in real-time project management, which is the industrial scenario that has been developed in this research work. The theoretical foundation that has been realized in this scenario is the behavioral modeling formulation as presented in section 6.1.3. The final scenario i.e. *scenario 3* presents the evolutionary model by extending the scenario 2 in the scope of real-time project plan optimization in the scope of construction project. The theoretical foundation that has been applied in this scenario is based on the formulation provided in sections 6.1.1 and 6.1.5. In the process of the development of all these scenarios, the design methodology and technologies as defined in sections 6.2 and 0 respectively have been utilized.

7.1. Scenario 1: Sensing Framework – Support Scenario for scenario 2

Sensors are increasingly becoming important in developing new forms of systems ranging from personalized mobile apps to enterprise systems for manufacturing. It has also been highlighted in EC report [264], that there is a need to decentralise intelligence, moving to a scenario where the enterprise is seen as a smart complex entity capable of sensing and reacting to (business) stimuli. At the same time having a seamless interface with the environment is an important necessity for evolutionary CPS.

Combining data from different types of sensors such as cameras, microphones, inertial sensors, beacons (e.g., GPS) and proximity-based sensors can greatly improve the accuracy and reliability of monitoring a particular situation. In order to fully exploit the capabilities of sensors, sensor networks and smart objects, it's necessary to formulate methodology to uniformly access, aggregate and process data collected from different sources. In other words, it's an important functional requirement to provide a solution where data sources can be seamlessly integrated and at the same time abstract the data source so that data consumers can consume data in a uniform standard way, without having to understand source details. Abstraction and sensor fusion allows designers to create virtual sensors that bridge what can be measured to what developers want to detect without having to realize new sensor as physical device. At the same time sensor fusion processes multiple sensor measurements of the same event at the same time thus making it easier to separate the "real" data from noise and sensor errors and use the output of the

virtual sensor as final result, thus increasing reliability of detected event. For example, a gyroscope with 100°/second bias appears to be spinning wildly when the device is at rest. Sensor fusion can zero-out this bias through comparison with the accelerometer and magnetometer data. Let's, say this new combination as motion sensor, and this allows designers to use such a combined effect of gyroscope, accelerometer, magnetometer and eventually gaining greater flexibility in component sourcing. Thus, when done correctly, there is no loss of responsiveness and this can overcome the shortcomings of individual sensors and provide useful, reliable results.

In this scenario we present the research and practical work that has been done to realize a flexible data collection framework for collecting data from heterogeneous sources via an abstraction layer over the physical sensors. At the same time the framework allows creation of complex virtual sensors by combining any numbers of virtual sensors, as discussed in section 6.1.4.

7.1.1. *SENSING FRAMEWORK*

The most important abstractions in Sensing Framework (SF) are abstraction over physical device protocols (specifically communication protocol) and abstraction over physical devices properties. The communication protocols abstraction allows seamless integration of different devices that have their own communication protocols while device properties allows creation of uniform way for identification, authorization and data access from different sources. Abstraction over protocol is achieved through protocol adapters and abstraction over device properties is achieved through virtual sensors discussed in section 7.1.1.2.

7.1.1.1. SENSORIAL MODEL INSTANTIATION

Sensorial model is based on the definition provided in sub-section 6.1.4. It is important to note that in the scope of this implementation, reinforcement sensing has not been implemented. In order to model the sensorial model let us consider three sensors i.e. 2 temperature sensors and one camera. The scenario being modelled is monitoring fire in the room and alerting where fire like situation is detected. The condition used for detecting the fire is if both the sensors measure same temperature for three time stamps, then scene needs to be captured and warning needs to be generated and fire alarm needs to be activated.

By following the sensorial model, sensors can be defined with a simple example as explained below.

Let's consider the values for *DataType* = {Integer, Float, Blob, String, Composite}. Now the sensors can be defined as:

S_{Temp_1}=<D=<Integer, {36, 36.5, 37, 36.2, 36.5}>, C=http://www.owl-ontologies.com/Ontology1268158035.owl#TemperatureSensor, C' ={Protocol=Serial, {Location=" LATITUDE=46.520000, LONGITUDE=6.565000}}>

S_{Temp_2}=<D=<Integer, {46, 36.5, 39, 40, 38}>, C=http://www.owl-ontologies.com/Ontology1268158035.owl#TemperatureSensor, C' = {Protocol=Serial, {Location=" LATITUDE=56.520000, LONGITUDE=16.565000}}>

S_{Cam_1}=<D=<Blob, {Cam1_1.jpeg, Cam1_2.jpeg, Cam1_3.jpeg, Cam1_4.jpeg }>, C=http://www.owl-ontologies.com/Ontology1268158035.owl#Camera, C' = {Protocol=zigbee, {Location=" LATITUDE=36.520000, LONGITUDE=10.565000}}>

Now VS can be modelled as follows:

$VS_{RoomMonitor}=S_{Temp\_1}\odot S_{Temp\_2} \odot S_{Cam\_1,}$ where,

$\odot$ is s.t. for i=1...n $D_{VSRoomMonitor}$ <Composite, $\{(x_i,y_i,z_i\}$ where $x_i \in S_{Temp\_1}$, $y_i \in S_{Temp\_2}$, , $y_i \in S_{Cam\_1,}$ where composite data type is the aggregation of the data type of sensors is defined by the data type of $S_{Temp\_1,}$ S_{Temp_2} and S_{Cam_1}.

So we get the data for, $D_{VSRoomMonitor}$ ={(36,46, Cam1_1.jpeg),(36.5, 36.5, Cam1_2.jpeg),(37,40, Cam1_3.jpeg),(36.2,70),(36.5,38, Cam1_3.jpeg)}

$VS_{Warning} = VS_{RoomMonitor}$ s.t. for i=1...n $VS_{Warning} = \{(z_i)\}$ where $z_i \in D_{VSRoomMonitor}$ and $x_i = y_i$ and $x_{i-1} = y_{i-1}$ and $x_{i-2} = y_{i-2}$

Which gives us $VS_{Warning}$= {Cam1_2.jpeg}

Actuator i.e. FireAlaram can be defined exactly as above except the action *A*, where $A_{out}(k)=f(VS_{Warning})$ *s.t.* $VS_{Warning}$!= *Empty*, $f(VS_{Warning})=Ring$.

From this instantiation, it's clear that we can model the system's sensorial model consisting of sensors, virtual sensors and actuators and corresponding actions by detecting various events. Note that in this example the timestamp is shown because it's not a model time parameter but a auto generated run-time parameter.

7.1.1.2. TECHNICAL INSTANTIATION

In this sub-section is provided the details of the technical instantiation of the sensing framework, which requires providing implementation of different functionalities to achieve three important goals:

- Allow the virtualization (i.e. representation, creation and management) of different sensors, virtual sensors and actuators (detailed in Virtual sensor specification)
- Allow plug and play like functionality to integrate devices with heterogeneous data and protocol standards. (detailed in Protocol adaptation)

- And allow continuous data collection and processing (detailed in Data Stream Processing in SF (Temporal Aspect) and Data Stream Processing in SF (Data Fusion Aspect)).

Virtual sensor specification

The key abstraction in SF is the virtual sensor (VS), which abstracts from implementation details of access to sensor data and correspond either to a data stream received directly from sensors or to a data stream derived from other virtual sensors. As stated before, VS can any kind of data producer, for example, a real sensor or any combination of virtual sensors. It needs to be stressed that a virtual sensor may have any number of input data streams and produces exactly one output data stream based on the input data streams and arbitrary local processing. The specification of a virtual sensor needs to provide all necessary information required for deploying and using it, which includes:

- metadata used for identification and discovery;
- the structure and properties of the data streams which the virtual sensor consumes and produces;
- a declarative statement for data stream processing performed in the virtual sensor;
- Functional properties related to stream quality management, persistency, error handling, life-cycle management, and physical deployment.

In order to support rapid deployment of virtual sensors, these properties are provided in a declarative deployment descriptor file with pre-defined schema. This functionality of creation and management of VS is provided by virtualization manager component (detailed in section 7.1.3). Important methods provided by this component are:

VirtualSensor(DataType[] inputs, DataType result) -Constructor for the VirtualSensor

query(ResultListener r) -Sends a one-time query to the VirtualSensor. The result listener receives results from this query.

register(ResultListener r, int reqfreq) -Registers a persistent query on the VirtualSensor. The request frequency reqfreq indicates how often the application demands the data value from the virtual sensor (i.e. window size). The method returns a receipt that can be used to cancel the registration when desired.

deregister(int receipt) -Stops the registered query referenced by the receipt

Protocol adaptation

Protocol Adaptation deals with bridging communication between different types of devices by hiding the device specific communication protocol requirements. In the CPS paradigm, devices to be used devices can be IP-based devices, that communicates using the IP stack (IPv4 or IPv6), or "legacy devices", meaning devices communicating using non-IP based protocols, for instance ZigBee, or Z-Wave. But, as explained in section

6.2.2.1 CPS requires having compatibility with NGN protocols. So, the Protocol Adapter receives these device specific protocols and translates them to a uniform internal API, which will provide access and management functionalities over all the connected devices. Figure 7-2 shows the internal components forming the generic protocol adapter.

Base Device Driver: The base device driver is the low-level API for legacy devices (i.e. an implementation of the device specific protocol stack). Base Drivers handle device discovery and access to sensor and actuator resources in a protocol specific way. For instance, the ZigBee Base Device Driver is based on the Network Device Gateway Specification defined by the ZigBee Alliance [265]. The functional operations to be performed by these drivers are: operations to read and write attributes, and configure and report events; macro operations for network and service discovery; endpoint management; flexible start-up and network join operations; bi-directional communication mechanisms devices and corresponding protocol adapter.

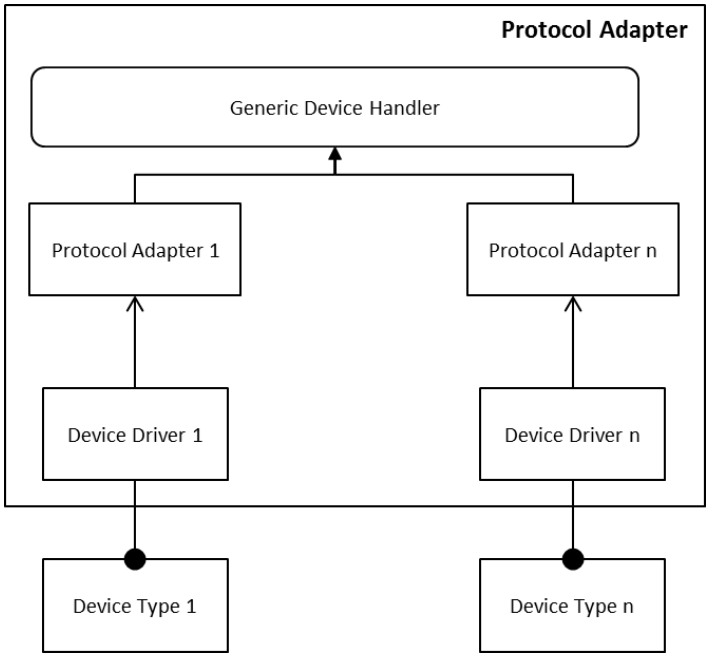


Figure 7-2 Protocol Adapter Internal architecture

Protocol Adapter: A Protocol Adapter is the glue between a base driver and the Generic Device Handler. Protocol adapter is implemented to translate device protocol to NGN protocol. Now, the higher layer components like generic device handler can access devices to collect data or provide action in a uniform way. Therefore, a Protocol Adapter is necessary for each protocol that the Base Drivers support. Whether the protocol is standardized or proprietary does not matter as far as the Base Driver is available and the Protocol Adapter is implemented on top.

Generic Device Handler: The Generic Device Handler (GDH) provides high-level, protocol agnostic functionalities and APIs towards the data consumers. GDH uses service schemas which are XML-files that describe the supported resources, i.e. the application profiles. This schema based approach makes it possible to cover a wide range of

applications spanning from home automation, to media, to health care. The GDA defines two main data structures:

Device: it represents the sensor/actuator and

Service: it represents a set of functionalities provided by the Device.

Some important methods exposed by GDH are:

device.getService(<name of service>) – used to get the services associated to a specific device,

service.getProperties() – used to get the list of properties (i.e. attributes) implemented by a specific service implementation,

service.getAction(<name of action>) – used to get a single action (i.e. command) implemented by a specific service implementation.

This way, we can achieve a protocol agnostic solution for seamlessly using different types of devices. Note that addition of new device type i.e. having unsupported protocol requires the need to implement both the Base Device Driver and Protocol Adapter and integrate them in the SF container.

Data Stream Processing in SF (Temporal Aspect)

Central concept in data processing in SF is the time model as it defines the temporal semantics of data and thus determines the design and implementation of a system. In most of the existing stream processing systems a global reference time is used as the basis for their temporal semantics because they were designed for centralized architectures in the first place. But in the CPS domain, temporal semantics is important and often the data sources are distributed. In order to address this issue SF provides essential blocks for dealing with time, at the same time leaves temporal semantics largely to applications allowing them to express and satisfy their specific, largely varying requirements. Thus in SF a data stream is a set of timestamped relations, i.e., each element of the data stream consists of a set of tuples. The order of the data stream is derived from the ordering of the timestamps. In order to achieve this SF provides service that performs following functionalities:

- Implicit management of a timestamp attribute (TIMEID) for all the data streams collected or processed by sensors or virtual sensors.
    - In case of sensor, its timed at the arrival of the data stream
    - In case of VS, data stream is always timed at the arrival of a data stream element from ***one*** of its input streams.
    - In distributed scenario if any one of the arriving data steam doesn't have its own time stamp, then the input stream is timestamped using the local clock of the SF, else it remains unchanged.
    - All distributed instances of SF follow synchronized clock for time-stamp.

- A windowing mechanism which allows the user to define count- or time-based windows on data streams.
    - Windowing can be used to limit the amount of data that needs to be stored for processing.
    - Windows can be defined using absolute, landmark, or sliding intervals. – The lifetime of data streams and queries can be bounded such that they only consume resources when actually active. Lifetimes can be specified in terms of explicit start and end times, start time and duration.

In this way it is always possible to trace the temporal history of data stream elements throughout the processing history and at the same time control the processing window to extract only necessary streams. This allows observation of the physical world, in which network and processing delays are inherent properties of the observation process which cannot be made transparent by abstraction.

Data Stream Processing in SF (Data Fusion Aspect)

In this section we will discuss the data stream processing in SF that explains how the virtual instance of sensors and also the virtual sensors are instantiated, which is as depicted in Figure 7-3 .

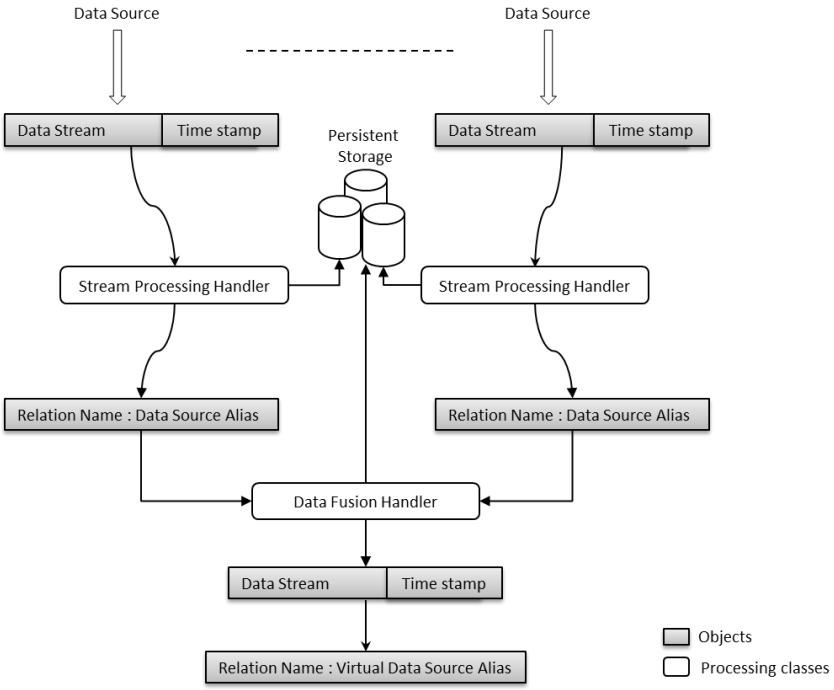


Figure 7-3 Data stream processing in SF

The data stream processing starts with the generation of a new tuple is with the data steam and respective timestamp. This is then passed to the stream processing handler that processes the data with operations like windowing, filtering etc. The generated pre-processed data stream is stored in persistent storage for historic purpose. The use of

persistent storage is also important in this stage to enable traceability of events that will be deduced by the other components in the stream processing pipeline. The result of stream processing handler is used to instant the virtual instance of the data source, which is identified by the data source alias name. This data source alias now can be queried by any data consumer to receive the necessary data.

Now, the virtual instances of data sources identified by data source alias names are used for instantiation of virtual sensors. This task is performed by Data fusion handler, which uses the defined configuration of the VS to select the necessary data streams and apply data fusion rule to merge data from individual data streams.

In order to specify data stream processing as described a suitable language is needed. A number of proposals exist already, such as in the Aurora project[6] users can compose stream relationships and construct queries in a graphical representation which is then used as input for the query planner. While the STREAM[7] project proposed the Continuous Query Language (CQL) which extends standard SQL syntax with new constructs for temporal semantics and defines a mapping between streams and relations. Similarly, in Cougar[8] an extended version of SQL is used, modeling temporal characteristics in the language itself. While the TelegraphCQ project[9] suggested StreaQuel language which tries to isolate temporal semantics from the query language through external definitions in a C-like syntax. For example, for specifying a sliding window for a query a for-loop is used. The actual query is then formulated in an SQL-like syntax. In SF, we have followed the similar approach and separate time-related constructs from the actual query. Temporal semantics e.g., the window size are provided in the virtual sensor specification, while data processing is specified in a subset of SQL. The main advantage of using SQL is the well adoption of SQL decreasing learnability burden on end users and easy adoption of query optimization and planning techniques can be directly applied.

7.1.2. *USE-CASE*

The use case that has been used for testing this framework is based on the scenarios described in our publication [266] and [267], which provides a discussion on personal health information management and continuous data collection in manufacturing industries by utilizing different types of sensors respectively.

[6] http://www.cs.brown.edu/research/aurora/

[7] http://www-db.stanford.edu/stream/

[8] http://www.cs.cornell.edu/database/cougar/

[9] http://telegraph.cs.berkeley.edu/

In the first scenario i.e. personal health information management working principles of the brain and nervous system is taken as the model. It is considered the way the brain and the nervous system permanently monitor a person's body. Information is acquired by our sensorial organs, is perceived and interpreted by the brain using existing knowledge, gained by learning and experience. The reasoning is based on a combination of different inputs, acquired by sensorial organs [268], which generate better assessment. With the use of different sensorial systems it is possible to monitor health status according to the context, and trigger events to inform or warn actions to be executed. So, the scenario under consideration is simulation of the way on how brain reacts to different types of stimuli from the real world environment.

While in the second scenario, it is motivated to provide solutions towards reactive manufacturing systems based on the real-time data. Continuous data collection from heterogeneous sources and infusion of such data into suitable processes can enable almost real-time reactive systems. In the context of today's manufacturing plants automated data collection is common, although often complex and difficult to synchronize and maintain. Extension on enterprise systems, by following right data integration can change manufacturing control processes. It has also been identified in 'Industry 4.0 -The new industrial revolution' [269] that new IT systems will be built around machines, storage systems and supplies. In this concept, data is gathered from suppliers, customers and the company itself and evaluated before being linked up with real production. The latter is increasingly using new technologies so that production processes are fine-tuned, adjusted or set up differently in real time[269].

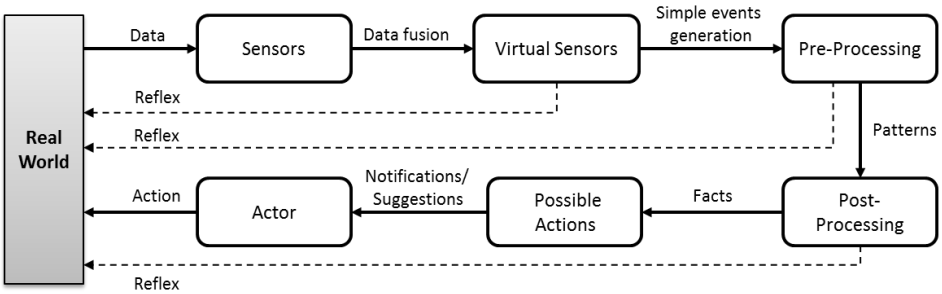


Figure 7-4 Data flow in stimuli based system

On, the whole both of these systems require a framework which allows seam data collection and integration from the data source without increasing added load over the existing enterprise or personal system. Figure 7-4 shows the data processing flow in the considered scenario. In the first step data is captured from the real world by various types of sensors or from other service providers. Thus collected data act as measurement of the parameter under consideration like temperature, humidity etc. In the next step, data infusion is performed leading towards more meaningful state like the temperature and humidity of an object at a particular instant, which leads to the generation of virtual sensors (VS). VSs will have data infusion rules such as if the temperature > x and humidity < y then activate VS. These are useful for generating simple instances of events, with multiple parameters, thus allowing more understanding of the real world and may

lead to reflex-like reactions as appends in the human body. The next step termed “Pre-Processing” is used for pattern extraction and will remove the unwanted data from the collected samples of instances thus providing filter over useful data. The patterns detected in this phase are passed to for post-processing, where domain knowledge and rules, are applied over the collected samples to generate facts. Interpretation of facts leads to generation of actions to be taken, over which actors can decide the follow-up procedures.

In the processing flow diagram, we also see that in each stage actions are generated, which we term as “reflex”- immediate response to the state of the real world. In order to act over the reflexes users should use their own knowledge and understanding of the overall scenario. The final actions are suggested based on wider processing of the scenario thus allowing users to respond with lower level of understanding. It is important to define some important concepts for further understanding of the methodology.

7.1.3. *TECHNOLOGY IMPLEMENTATION*

One implementation of SF has been provided in the scope of the FITMAN[10] project. The implementation is named FITMAN "Shop Floor Data Collection" Specific Enabler (SFDC)[11]. This technological is developed together with ATOS[12] to act as the middleware between the data producers in the shop floor and the data consumers. At the same time this SE also decouples the event producers from events consumers to provide flexibility in the processing of production data. Data producers (which can also be the producers of events) are integrated into the main system making use of smart object technologies like RFID and sensorial networks. Data consumers are various applications and services, which make use of various components of this implementation as necessity. The decoupled layers of technical solution is as shown in Figure 7-5.

Figure 7-5 Layers of technical solution of scenario 1

SFDC implementation, which is the implemented instance of SF contains two distinct components:

[10] http://www.fitman-fi.eu/

[11] http://catalogue.fitman.atosresearch.eu/enablers/shopfloor-data-collection/documentation

[12] http://atos.net/en-us/home/contact-us/spain.html

Fitman Tag&Trace [FitmanT&T]: This component is based on Fosstrak for data acquisition and is responsible for collection of data from objects in the shop floor with RFID tags to provide track and trace functionality for the shopflor.

Fitman Sensor Network [FitmanSN]: This component is based on results OpenIoT project[13] and is the middleware solution for rapid deployment and integration of heterogeneous wireless sensor networks.

We implemented the SF container in Java. The core implementation of the container is around 5,000 lines of Java code which encompasses the query processor, query manager, network manager, input stream manager, storage manager and life cycle manager in addition to the web interface. For deploying a virtual sensor a user has to specify the XML deployment descriptors. For enabling a new type of sensor or sensor network a Java-based base device driver implementation is required. The existing implementation can be obtained from FITMAN repository[14].

7.1.4. *OBSERVATIONS*

Research presented in this scenario has potential impact in many business aspects, some of which has been exploited in different smart and digital factory trials of Fitman and have been reported in the Fitman Experimentation report [270] and [271]. The use of sensing framework allowed industries to work with new business and functional scenarios that were enriched by real-time data. By the usage of the proposed Framework, we could integrate sensorial data and their interpretation into the existing business process, and eventually generating events, notifications and advices where applicable. It has been reported in FITMAN Smart-Digital-Virtual Factory Trials Experiences [272] that the SFDC (lighter version of SF) configured for FITMAN has significantly improved the overall efficiency for data collection and integration in existing business process of the industries like Consulgal[15]. At the same time the current implantation is being extended in the scope of the C2NET[16] project to adapt it to wider cases of manufacturing industries such as FLEXF - Flexefelina, SA[17] and AAMM - Antonio Abreu Metalomecanica, LDA[18].

[13] http://www.openiot.eu/

[14] https://sourceforge.net/projects/fitman-fi/files/Fitman%20Specific%20Enablers/Smart%20Factory/Shopfloor%20Data%20Collection/

[15] http://www.consulgal.pt/en/

[16] http://c2net-project.eu/home

[17] www.flexefelina.pt

[18] www.aametalomecanica.com

7.2. Scenario 2: Situational Awareness for Real Time project Management – Industrial Scenario for Validation

The interaction between a computing system and real world objects is becoming an issue of major importance in business. The ability to gather real-time information, generated from virtually any objects in the real world, will open up new paradigms for applications that range from personal use to business activities. At the same time it requires important research effort to address the acquisition and management of such information and harvest useful knowledge from such sensorial raw data. Various types of sensors are available that can be used to monitor subjects and instruments which generate a continuous flow of information which, in return, provide information services that are accurate and current i.e. almost in real-time. This information is very useful in complex real-world systems that need to perform timely reactions to selected environmental events. This scenario provides new paradigms for making use of information collected from surrounding environment, allowing integration of real-time information with legacy information system. This approach will increase awareness and support faster decision-making by the involved stakeholders.

Situation awareness, aims at determining the meaning of information about perceived events, and is the basis for making decisions in heterogeneous, highly-dynamic environments. Historically, the most common means of obtaining situational awareness would be to identify what information is required for the mission and then to proceed with search routines through printed material, data repositories, users' feedbacks, activity logs etc. This manual process is very time-consuming and may not produce timely results. As such, IT infrastructure and systems had to be developed to collect data and information from disparate sources. Adding to this would be necessary to process data into logical outputs, and push it to the user to make better and more informed decisions.

This validation scenario proposes a framework for project management with the capabilities to detect and gain insight into processes going on at the shop-floor, or at the site of project execution. In real world, at the manufacturing site, the on-going processes are influenced by various environmental factors. Those are, along with other uncontrollable constrains like the weather, the performance of workers and the activities of stakeholders involved in the processes. The proposed framework tends to integrate all these factors into a computing system to generate situational awareness to help the project supervisor to speed up the process of decision-making. It enables and encourages them to engage in mindful reflection of the project status and respond more critically to difficult or unexpected, circumstances. Following sections will provide the motivation for this research work along with the details of the proposed framework. An implementation detail of the framework, with a use-case scenario for project supervision in construction industry, is also provided in the following sections.

7.2.1. *USE-CASE*

This validation scenario proposes a framework for project management with the capabilities to detect and gain insight into processes going on at the shop-floor, or at the site of project execution. In real world, at the manufacturing site, the on-going processes are influenced by various environmental factors. Those are, along with other uncontrollable constrains like the weather, the performance of workers and the activities of stakeholders involved in the processes. The proposed framework tends to integrate all these factors into a computing system to generate situational awareness to help the project supervisor to speed up the process of decision-making. It enables and encourages them to engage in mindful reflection of the project status and respond more critically to difficult or unexpected, circumstances. Following sections will provide the motivation for this research work along with the details of the proposed framework. An implementation detail of the framework, with a use-case scenario for project supervision in construction industry, is also provided in the following sections.

Cyber physical system can be realized in the construction industry which is also one challenge being addressed by the Portugese trial in the scope of FITMAN project. As it has been suggested in [273], CPS in construction industry can be realized by the adoption of a 'system of systems' approach, which will be an important factor for improvement productivity and efficiency [274]. Since construction project involves direct interaction with the physical world (construction site), there will be a bi-direction interaction between the cyber world and physical world, which is as shown in Figure 7-6.

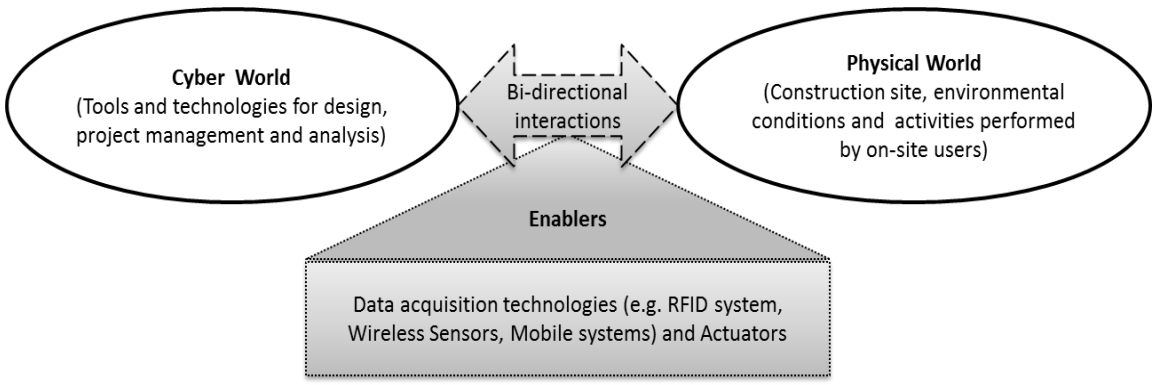


Figure 7-6 Bi-directional interactions between cyber world and the physical world in construction project

This is particularly important as it supports design-build and other integrated project delivery methods. This concept of integration is important to align the as-designed and as-built scenarios of the construction project. It will be important for active monitoring and control of construction activities such that as changes in one of the world will be reflected in both the worlds, thus enabling efficient project delivery and decrease in the overhead caused by delayed decision making process.

Realization of a construction project as a CPS, will this offer opportunities for improving information handling, thus enhancing access to real-time information or communication

between the design, construction team and supervision team. With bidirectional interactions various stakeholders involved in the project can have access to the design parameters, construction activities, and environmental factors and make queries over as-is state of the project life cycle to receive status and feedbacks to time critical issues, thus reducing delays to projects.

7.2.2. *SITUATIONAL AWARENESS*

Situation Awareness (SA) provides the basis for increasing the quality of decisions by determining the meaning of information about the perceived events. SA originates from applications of cognitive sciences to the aviation and military domain, SA has been defined by Endsley [275] as ”the perception of elements in the environment within a volume of time and space, the comprehension of their meaning, and the projection of their status in the near future”. International Society for Information Fusion (ISIF)[19] provides a number of research works in the domain of achieving SA using computational models. One important data fusion model for achieving SA is discussed in [276] which is termed as JDL data fusion Model. The JDL Data Fusion Model has become a basis for a number of research works (e.g. e.g. [277], [278], [279]) that for explicitly defining SA.

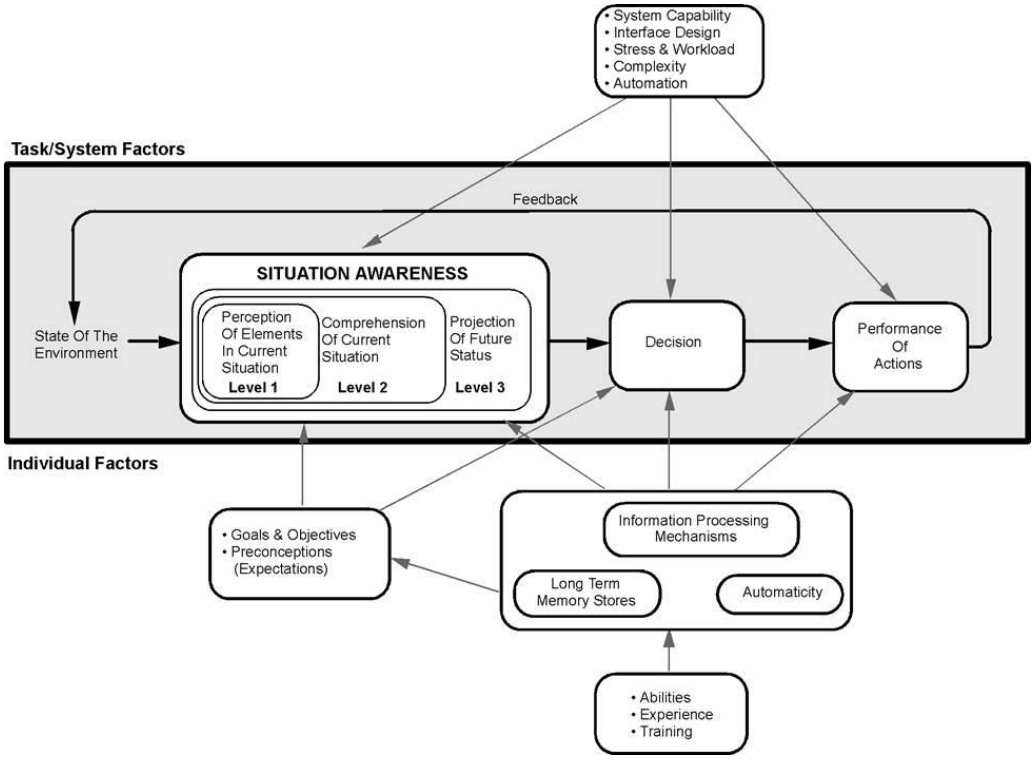


Figure 7-7 Endsley’s Situation Awareness Model

[19] http://www.isif.org/

Endsley [280] provides an alternative to the JDL model that provides a holistic viewpoint of Situation. This has two main parts: the core SA portion and the various factors affecting Situation Awareness. The core portion follows Endsley's [275] proposition that Situation Awareness has three levels of mental representation: perception, comprehension, and projection. The second and much more elaborate part describes in detail the various factors affecting SA. The three levels of Situation Awareness as proposed by Endsley are summarized in Figure 7-7.

Based on the definition of the SA, it can be observed that integration of real-time information provides huge added value on SA frameworks. Situation awareness involves the real-time processing of information, based on various events occurring on the shop-floor, from an evolving situation in an attempt to understand what is happening. [281], situation awareness primarily comes down to identifying higher-order relations that come into being within a situation, as defined by the user's goals or objectives. By higher-order relations it means relations involving multiple objects. Thus the analytical process of establishing situation awareness necessarily involves fusion of data, from multiple heterogeneous sources, to produce new enhanced information.

7.2.2.1. SITUATIONAL AWARENESS IN PROJECT MANAGEMENT

As identified in Factories Of The Future (FoF) roadmap , the major challenges that manufacturing companies face today are: the growing complexity of their processes and supply networks, cost pressures, growing user and customer expectations for quality, speed, customized products, worker's safety and assistance. Manufacturing is evolving from being perceived as a production-centred operation to a human-centred business with a greater emphasis on workers, suppliers and customers being in the loop [282].

The foremost requirement for addressing these challenges is collaboration. In collaborative manufacturing, ICT will support a constant feedback loop without media breaks between product designers, engineers, state-of-the-art production facilities and customers [283]. Collaboration solves an important aspect of the problem but, at the same time, increases the complexity of the overall process with the increase in the number of stakeholders and diverse locations for activities. Project management within diverse groups, often located in remote locations, will face a number of deviations in the original project plan.

So, it is very important to identify the events that occur at various phases of project execution and provide intelligent decision support, based on the applicable business rules. The almost real-time detection and processing of events will help the supervisors and project managers (and other stakeholders where applicable) to reduce decision-making process and eventually save time and money. At the same time, the understanding of the global situation of the project execution will be useful for identification of the possible deviation, on the following activities, and for the mitigation of future risks.

This research work is thus, motivated towards a unified framework to generate situational awareness by integrating IoT technologies into the legacy systems. It will be an important block towards the factories of future that are more responsive, regarding efficient collaboration between stakeholders and, deeply integrated into the real-world models. This framework provides the capability to the project supervisor to respond quickly, and efficiently, to the abnormalities detected in the project plan. In this research work, authors are motivated to provide a framework that will add value to the project manager procedures with an efficient way to perceive situations resulting from odd events, occurring at various stages of a project execution. It will also allow the extraction important knowledge that can be used for future project planning.

7.2.3. *SITUATIONAL-AWARENESS IN CONSTRUCTION INDUSTRY: INDUSTRIAL SCENARIO*

Construction sector is selected as the reference use-case scenario for the implementation and testing of the proposed framework. FITMAN[20] deliverable on business requirements of the project [284], has clearly indicated the need of efficient decision-making process in the construction industry. Management of complex construction projects involves planning and monitoring tasks and resources for a single project. However, programme planning is the planning and monitoring of tasks and resources across a portfolio of projects.

Construction programme supervision involves planning at the strategic level where tasks are delegated to the project manager. The project manager on the other hand, is involved in planning individual project, where task are delegate to departmental manager or team leader or some other user with defined role. All the users with varying roles can belong to different organizations or stakeholders of the overall project. Therefore, the situational-aware programme planning, monitoring and control application may be designed, not just to be the project and programme management viewing tool, but as an application that intelligently update tasks of data fields (such as % complete, resources name, planned start and planned finish for tasks and milestones).

In order to model the construction supervisor scenario, the modelling of the project is as shown in Figure 7-8. This model gives a global view of the project. ‘Activity’ represents all individual tasks that have to be performed in the project. Each tasks performed at various stages can lead to the generation of events. Those will be used to generate situations, based on the applicable business rules for the particular tasks, by including the global rules that are applicable in the overall project. Note that input to any activity is, not only the necessary information for the task but, also the business rules.

[20] http://www.fitman-fi.eu/

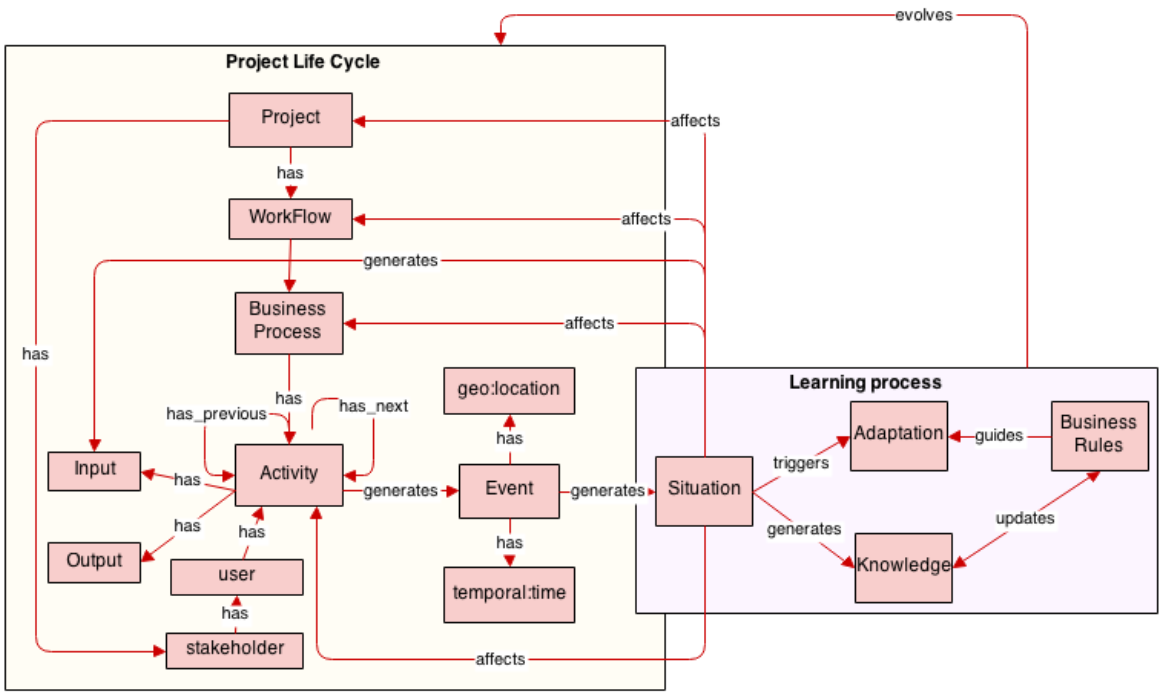


Figure 7-8 Meta Model of a Project including learning over the processes

Based on this model, events that arise at each phase can be traced back to the corresponding project, and the stakeholders involved, and the global project plan. This will help the supervisor to correctly access the situation and take corrective decisions were necessary. He can eventually use the knowledge generated also for future project planning. In order to further define the situation, this research work have taken into the reference model presented in [285] and is as shown in Figure 7-9. The central entity of this model is Situation, which is referenced in the model of the project. Instances of this class have properties of *relevantIndividual*, *focalIndividual*, *relevantRelation*, and other applicable properties.

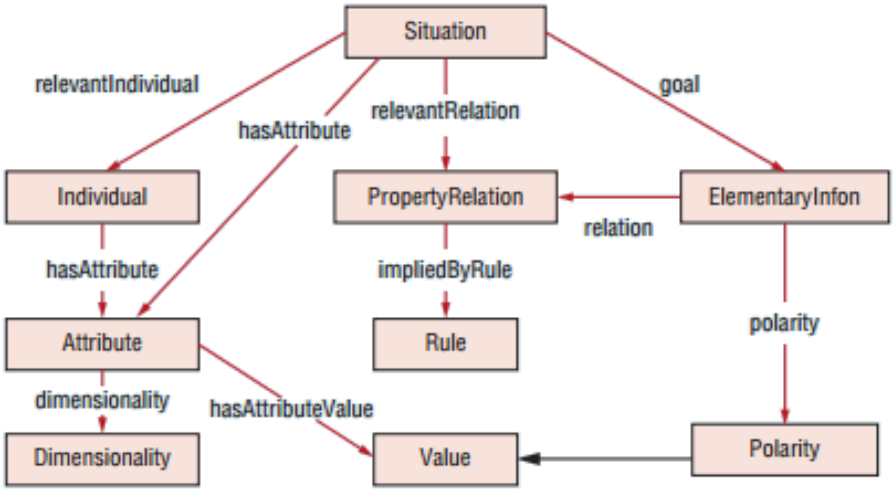


Figure 7-9 Top level of the Situation Theory Ontology

The other classes include Individual (individuals involved in a particular situation), *Attribute* (attributes of individuals or situations), *PropertyRelation* (relations that are relevant to the situation), and *Rule* (conditions that need to be true for a particular relational tuple in a given situation, and the inferences that are drawn when the conditions hold). Infon represents queries, or goals— statements that give focus to a particular situation. An example of such a statement could be a query, "*Is test results from operation BAX-SAB-OP15 safe*?" Polarity is a special kind of value; it can be either 1 (meaning the infon is satisfied) or 0 (not satisfied), corresponding to whether statement represented by the Elementary- Infon is true or false, respectively. The notions of ElementaryInfon and Polarity come from Jon Barwise's situation theory [286].

A simple use case scenario within the scope of construction project supervision is handling the concreting operations. Concreting is a critical activity and has a huge impact on the overall project planning and maintenance. This impact reflects on the timing of operations, reliability and, ultimately, in costs. One of the business processes in this scenario is *"samples collection and test"*.

The concrete is manufactured with the characteristics defined in the design documents and is transported to the construction site (in this case a dam) by truck. Upon arrival of a truck, a sample is collected for the slump test, which is carried out in the presence of an element of the Supervision team. Thus collected samples have to undergo a series of test viz. slump test and laboratory test. Results from these test operations are critical and must comply with all the standards and regulations.

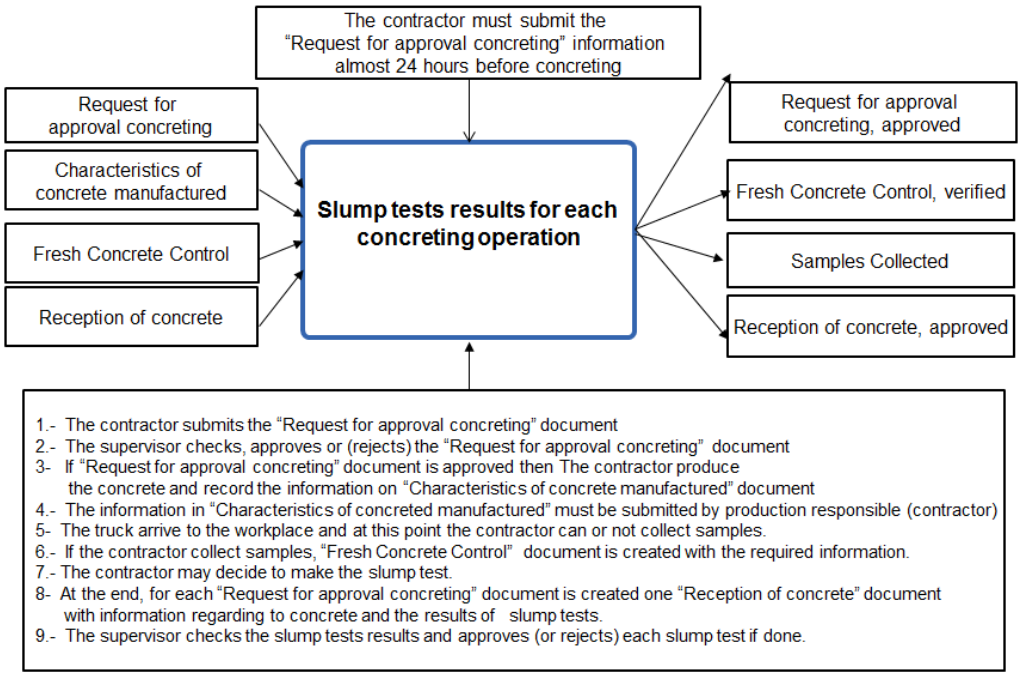


Figure 7-10 Details of slump test activity

Incompliance of the results, will affect the future operations and the actions that supervising authority need to take in order to avoid any faults in the construction project. One of the task i.e. “Slump test and results” is as described in Figure 7-10. This activity is mapped to the model, as described in Figure 7-8 in the reference implementation of this work. It can be easily observed, with this simple activity, that a number of events can occur which can have a long-term impact on the project. In face of those facts it becomes critical for the supervisor to have, an almost real-time, situational access of the project at the site.

7.2.4. *BEHAVIOURAL MODEL INSTANTIATION*

Situational awareness in the scope of construction project can be modelled with the instantiation of behavioural model as described in section 6.1.3. In order to model the situation model let us consider the business process of slump test activity as defined in the previous section. The slum test business process is preceded by other business processes such as concrete definition, concreting plan, concreting operation etc. which provide the pre-conditions for this activity. Also, it’s important to note that slump test activity is highly dependent on the environmental conditions. Now in order to define the overall situations let’s consider *World* which is composed of tasks, human resources (R), machinery resources (M), location (x, y), plan, do(action), actions are performed or deferred, some actions are not completed, plan can fail/succeed, supervisor(member of human resource) can correct the plan for deviation.

The possible actions (A) can be defined as:

- Perform(R, t): R performs the task t.
- Validate(R,t): R validates the task t
- Approve (R, t): R approves the task t
- Reject (R, t): R rejects the task t

Fluents that define the properties of the world can be defined as:

- is_performing (t,R,s) : R is performing the task t at state s
- is_beingUsed (t,M,s): M is being utilized in task t at state s

Now, the truth axioms for these fluents can be defined as:

- is_beingUsed (task_1, Machine_1, S_0): FALSE ; Initially the machine is not allocated for any task.
- is_performing(task_1,Resource_1, Perform(Resource_1, task_1): TRUE; Resources 1 is performing task 1 when the action Perform is true.
- is_performing (task_1, Resource_1, Not_Raining): TRUE This means that the task_1 can be in performing state by Resource_1 if its Not_Raining

Note that Not_Raining can be defined based on the sensorial model definition as defined in section 7.1.1.1.

Now the action preconditions can be defined as:

- Poss(a, s): It's possible to perform action a in state s.
- Poss (Perform(Machine_1, Task_2), NotBroken (Machine_1)): Its possible for Machine_1 to perform Task_2 if Machin_1 is assigned to perform Task_2 and its not broken.
- Poss(Perform(Resource_1, Task_2), Approve (Resource_2, Task_1),): Resource_1 can perform the Task_2 in the situation such that Task_1 performed by Resource_2 has been approved. It signifies that completion and approval of Task_1 is the precondition for the start of the Task_2, and that Task_1 precedes Task_2 in the plan.
- Poss(Validate (Task_1, Resource_1)) ↔ (∀z ¬ is_performing (Task_1,Resource_1,State_1) ∧¬Failed(Plan)): Task_1 possible to be validated if and only if the task is not under the is_performing situation and the overall plan hasn't failed.

Now, the successor state axioms can be defined as:

- Poss(Task_1,Completed) → [Validate(Task_1,Poss(a,s)) ↔ (a= Perform (Task_1, Resource_1) ∧ Approve(Task_1, Resource_2))]; Task_1 can move the completed state if the task has been performed and validated.
- Poss(Task_2, Start) → Poss(Perform(Resource_1, Task_2), Approve (Resource_2, Task_1),) ∨ (deferred (Task_1, S) ∧ a ≠ Failed(Plan, S))]: Task_2 can start if Task_1 has been approved or Task_1 has been deferred and the overall plan hasn't failed yet in state S.

Based on these axioms the goal of the overall behaviour can be defined as:

- Goal |= ∃s Do(CompletePlan, S_0, s) where s =do(Perform(R, t),do(Validate(R,t), do(Approve (R, t), Reject (R, t),S_0))))

It implies that the goal is to complete the plan, the initial state is S_0 and the actions that can be followed are performed, which is then validated which can be either approved or rejected. And it clear that the transition from one task to another is defined by the precondition and successor axioms as defined above.

Note, that this is just a small portion of the axioms that are needed to be defined and implemented in the process of defining situational behaviour of the scenario under consideration. And, it's also important to note that in the scope of the implementation in this research work, the above mentioned axioms are represented in XML, thus not providing a well-defined semantics, thus providing an important future work. The axioms

are in second order, thus, requires a methodology for translation to first order so that they can be represented with semantic web languages such as RDF and OWL.

7.2.5. *TECHNOLOGICAL IMPLEMENTATION FOR SITUATIONAL-AWARENESS*

This module is defined based on the model as explained in Figure 7-11, with a clear distinction between the perception and comprehension layers, which are the important layers of the 4 layered situational awareness reference model. The detail of this module is as shown in Figure 7-11.

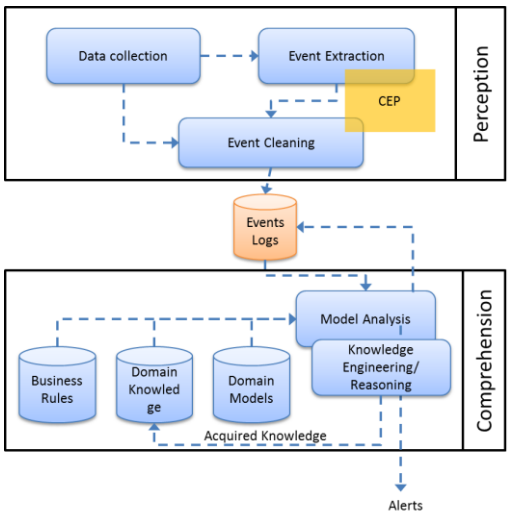


Figure 7-11 Details of the situational awareness module

Perception layer is responsible for collection and processing of data. This layer interfaces with the lower layer, to receive the data from the shop-floor. The data thus received is processed based on the CEP rules and the domain of interest. Based on the different types of data collected, different methodologies or algorithms will be integrated in this layer in order to extract events from the collected data.

The generated events are stored in the events log database. As the events database is updated, Model Analysis, Knowledge engineering and Reasoning tools are used to determine if the events are critical and to define evidence from them, with necessary contextual information. This layer is thus responsible for actual generation of situational alerts. This layer will make use of the domain model, business rules and domain knowledge. This portion of the process defines the “Comprehension” portion of the model. Here the knowledge acquired will be integrated with the knowledge base. This module thus combines the knowledge provided by the systems analyst with the obtained evidence (or perception) in order to provide comprehension or understanding of the situation. Note that this is clearly supported by the generic architecture presented in section 6.2.2.1. The situational awareness module is custom implementation to handle situational awareness and is integrated in the overall system via. Custom

Application/Service Integration Framework. Hence forth, this module will utilize the data collection component will work together with the lower layers of the

7.2.5.1. TECHNOLOGY ADOPTION

Implementation of the system is based on the reference architecture [284] proposed within the scope of FITMAN project. This work adopts services and enablers, being developed by projects, under the European FI-PP program like FI-WARE[21] and FITMAN, which provide enablers for data acquisition from the physical world (Internet of Things Service Enablement), complex event processing, context aware data handling, data analysis etc. Figure 7-12 shows the different layers of the technical solution. The behavioural model is used to model different actions and activities that will occur in the scope of the construction project. The sensing layer is the implementation as described in scenario 1. The SA layer is the situational layer implementation as descried in previous section. Based on these independent implementation, application has been developed which allowed construction supervisor to effectively monitor the concreting process with the automatic detection of events/situations to enhance the decision making process.

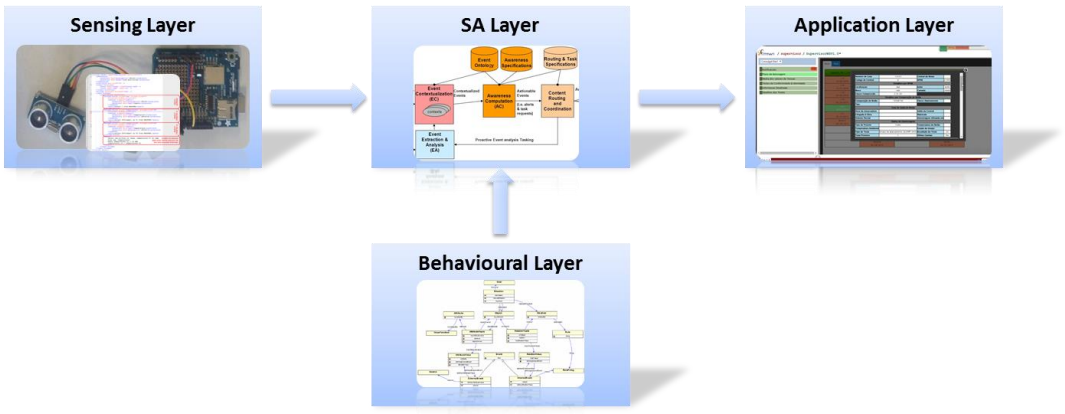


Figure 7-12 Layers of technical solution of scenario 2

Shop-floor Data Collection[22] (SFDC) specific enabler, which is the implementation as described in scenario 2 is used for integration of RFID tagged objects into the information system. In the production environment, SFD is deployed together with Secure Event Management[23] enabler, to provide a secure dispatch of events collected from the shop-

[21] http://www.fi-ware.org/

[22] http://catalogue.fitman.atosresearch.eu/enablers/shopfloor-data-collection

[23] http://catalogue.fitman.atosresearch.eu/enablers/secure-event-management

floor. CCEP Complex Event Processing (CEP)[24] and Publish/Subscribe Context Broker - Context Awareness Platform[25] implementations, from the FI-WARE project, are integrated into the information integration layer of the architecture, as presented in 6.2.2.1. CEP is used for analysis of event data, in real-time, and generate immediate insight to enable instant response to changing conditions. This implementation is thus used to react to situations rather than to single events. The situation is generated based on a series of events, which have occurred within a dynamic time window, called processing context. Context Broker enabler implements publication of context information by entities, referred as Context Producers, so that published context information becomes available to other entities, referred as Context Consumers Those are interested in processing the published context information. Both of these enablers are the important components that are used for integration with the legacy system by use-case specific implementations.

The implementation of situational awareness module makes use of formal knowledge representation and semantic web technologies. At the same time this module makes use of BigData Analysis[26] enabler to process huge amounts of previously stored data, in order to get relevant insights in scenarios where latency is not a highly relevant parameter. In this layer, integration of "Semantics of Business Vocabulary and Business Rules" (SBVR) standard [287] into the BPMN business process model, is also considered to provide a semantically enriched business modelling with formal grounding of business rules.

7.2.6. *OBSERVATIONS*

The implemented solution was used in real-case scenario of Consulgal, during the concreting operation at Estoril Grand Hotel. Sensors utilized for the experimentation were RFIDs, motion sensors and web cams. This was used to model a situation, where the supervisor could monitor the concreting operations from the remote locations and at the same time system generated event notifications regarding the start and end of the concreting operations by analysing the motion of the people at the work site. Also the RFID tags were used to trace the concrete samples collected for testing, which are collected by the personnel from the lab, which was remotely located.

[24] http://catalogue.fi-ware.org/enablers/complex-event-processing-cep-ibm-proactive-technology-online

[25] http://catalogue.fi-ware.org/enablers/publishsubscribe-context-broker-context-awareness-platform

[26] http://catalogue.fi-ware.org/enablers/bigdata-analysis-cosmos

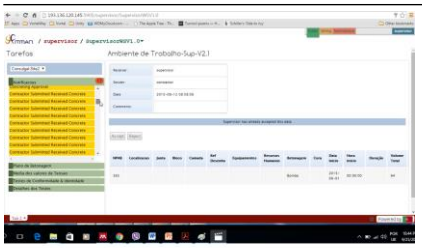


Figure 7-13 Notifications of events and/or situations

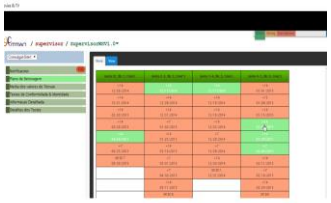


Figure 7-14 Details of detected situation (Concreting plan success/fail)

Figure 7-15 Details of the success/failure

The supervisor is alerted with events if the lab failed to perform test as planned and also when the tests are performed on time, so that the next activity in the project plan could be triggered. But, during this experimentation phase, even though the functional validation of the solution could be made with simple situations, the system was not used for large number of project cycles to observer adaption, learning and evolutionary behaviour.

7.3. Scenario 3 – Project plan Optimization with Evolutionary Algorithms- Enhancement of Industrial Scenario

One of the most challenging tasks of a project planner is to simultaneously minimize the total project cost and total project duration while considering issues related to optimal resource allocation and resource levelling. Therefore, project planners face complicated multivariate, Time-Cost-Resource Optimization (TCRO) problems that require time-cost-resource trade-off analysis. In this section, we present an EA approach to solve TCRO problems in construction project planning. Our objective is to create a superior optimization method than existing optimization algorithms to find better project schedule solutions with less total project costs, less total project durations, and less total variations of resource allocation. The solution presented in this section is based on the theoretical presented in section 6.1.5.

7.3.1. *Use-case*

The basic use-case is the same as described in section 7.2.1. But, in this scenario, is presented the experimentation of the basic scenario by utilizing knowledge injected EAs. In order to understand the concreting plan model refer Figure 7-16, which presents the details of the concreting plan that has been adopted from the FITMAN Consulgal's use-case. The basic model of concreting plan presents the Juntas (i.e. connection points between different blocks of the construction), Bloco (the blocks that are to be concreted) and layers (different layers of a block). It is important to note that different layers are concreted with different concreting specifications. For each cell is provided the date of

concreting and the approximate date after which the next layer can be concreted. In this concreting plan, the resources needed for the operation is not depicted though.

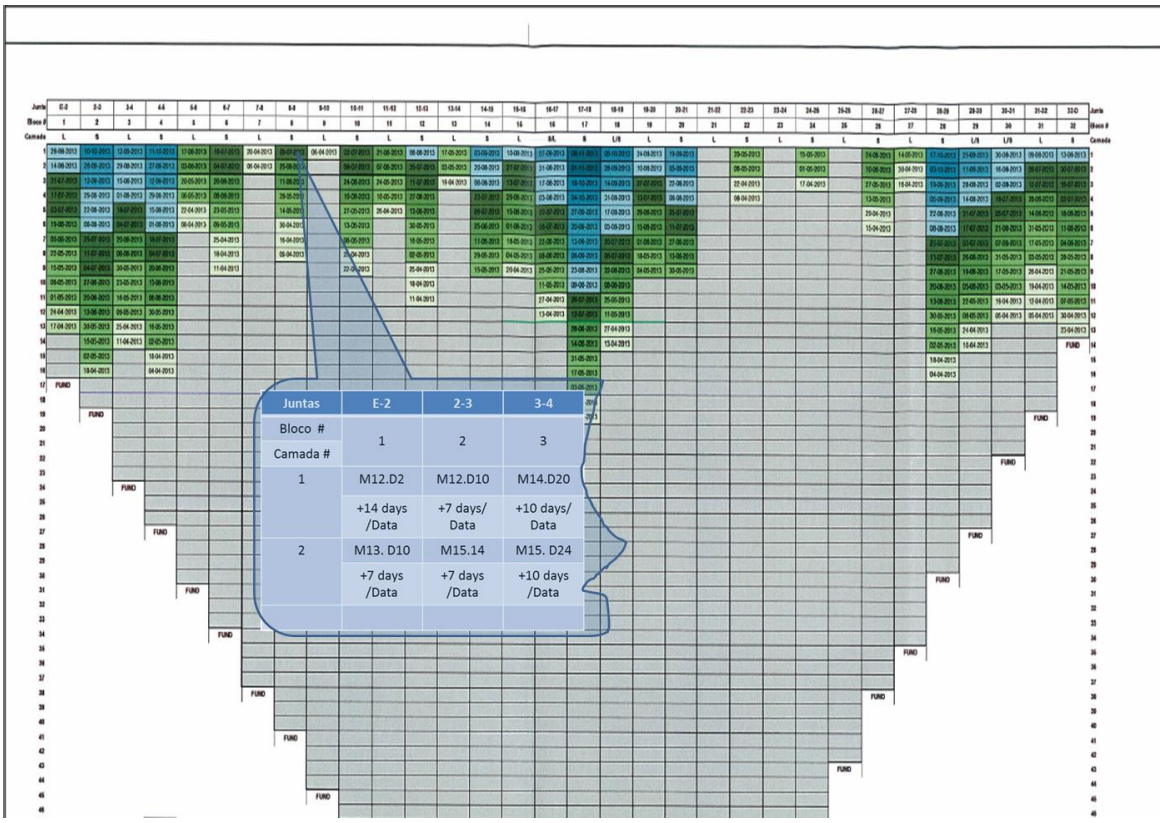


Figure 7-16 Concreting plan details

Based on this basic plan model followed in the concreting plan, the following sub-section formulates the detailed plan model that will be used for development of the evolutionary model for real-time optimization of the construction project plan.

7.3.2. *EA MODEL INSTANTIATION*

Evolutionary computational model in the scope of this technical implementation is defined based on the theoretical foundations presented in sections 6.1.1 and 6.1.5 by considering the industrial scenario of construction project management. The concreting plan under consideration is as defined in section 7.3.1. In the following sub-section is provided the details of the EA model starting with modelling individuals (Plan Model in section 7.3.2.1), modelling optimization criteria or fitness function (Plan Optimization Model in section 7.3.2.2) and Evolutionary operators model (Evolutionary Operators in section 7.3.2.3). It is important to note that the implementation of this scenario utilized both the solutions from the previous scenarios to capture the environmental factors and behavioral dynamics of the system.

7.3.2.1. PLAN MODEL

The Plan model was modelled to capture the most important characteristics of the planning activities. Plan model forms the "individual" of the EA model and collection plan models will act as the "population" for EA model. The plan is defined with following four attributes (also depicted in Figure 7-17):

- Sequence of Activities: array that contains sequence of the activities. The order of activities is an important characteristics to model the precedence of the activities, which are aligned to pre-conditions axioms in behavioural model.
- Starting Dates: For each activity is provided the starting date which is a vector respecting precedence constraints
- Ending Dates: For each activity is provided the ending date that contains the completion time of each activity.
- C_{opt}: For each plan model is provided the objective function.
- Wait Time: (Optional), For each activity that belongs to concreting operation, the waiting time is provided that important to define the starting dates for the proceeding activities of the same type.

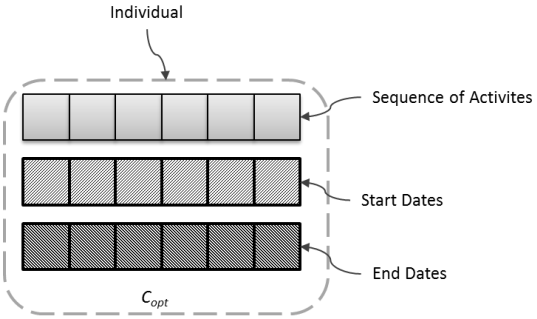


Figure 7-17 Plan model that acts as individual for EA

The collection of individuals as shown in Figure 7-17, forms the population for EA. Also, note that the model of the plan as provided above can be easily represented in OOP paradigm, which is the followed methodology in the technical solution. This structure is found within every generation of the GA.

In order to further extend the plan model we will make use of the ontology as defined in section 6.1.5. Our domain ontology formalizes the main concepts concerning our problem, such as tasks, resources, people, and locations and so on (see entities below *DomainEntity* in Figure 7-18). One of the essential components of the planning problem is *PlannedElement*. It is an event to be held at a given location, which should start at a desired time (*requestedDate*), and should last a known or estimated time (*duration*). There are many subtypes of *PlannedElement*, some of them are shown in Figure 7-18. Also note that the ontology that has been developed in this scenario also captures the model as presented in scenario 2. Some important entities are:

- *PlannedPersoneElement*: events related to an employee. For example, such employee shall be present at the site before the activity starts by let's say half an hour. On the same principle, there are also *PlannedResourceElement* events related to resources: the machine must be revised or checked for dysfunctionalities beforehand, etc.

Figure 7-18 Ontology with both the domain and EA entities

- *PlannedMachineElement*: an event that is not directly dependent on a machine or on a particular employee. The optimization algorithm will have to determine which machine operated iven by which employees will be assigned to the event. There are again two subtypes of *PlannedMachineElement*:
    - *BusinessTaskElement*: this is the most classical event: a task must occur somewhere. These events are paired within activities, which includes the collection of materials, the transport, deposited to another site, concreting operations etc. Of course, a task started by one person has to be closed by the same person. The optimization will obviously have to take this basic constraint into account. The *task* is associated to *PlannedElement* and not directly to a *BusinesTaskElement* since constraints related to the task may be associated with activities.

- *InternalTaskElement*: these are constraints that are not directly related to the task (e.g., fetching a document from office).

The assignment of the events to individual resources is the main goal of the optimization. A *PlanTimeLine* represents the list, ordered by increasing time, of the events supported by a given resource. A complete Planning is simply the set of the PlanTimeLine for all the resources.

Finally, the algorithm must take into account many additional constraints. For the sake of simplicity only two constraints are used in this scenario:

- *ResourceCost*: This includes the cost that has to be incurred for using certain type of resource for defined activity.
- *PersonnelCost*: Additional costs that might incur for specialized personnel to be used for the activity.

The ontology for Evolutionary Algorithms (EA) ontology provides generic, domain independent, entities and properties allowing defining what use will be made of entities and properties of the domain in the genetic algorithm. The EA ontology is therefore a characterization of the entities and properties of the domain ontology seen as individuals. The elements such as *EAEvaluationProperty*, *EAOptimizableProperty*, *EAParameterProperty* etc. form the part of the EAentities and are described in more details in section 6.1.5.

7.3.2.2. PLAN OPTIMIZATION MODEL

The project planning problem is not a single-objective optimization problem, but a multi-objective one, because optimal decisions need to be made in the presence of trade-offs between two or more, eventually conflicting, objectives. Multi-objective approaches appear clearly as a possibility to solve the problem after a careful analysis of the constraints coming from the underlying data model. Moreover, beyond the classical advantages of these approaches, they are efficient in limiting a concentrated convergence of the solutions in a small subset of the Pareto front, which is very interesting for knowledge intensive evolutionary algorithms. The algorithm that has been considered in the scope of this implementation is as described in Appendix A-section 11.3. To be able to define a set of objectives, we first have to present a macro-objective and then break it down into a series of micro-objectives. The macro-objective is defined as: *Based on a set of required elements, resources and employees, we have to generate a planning of itineraries that these resources will be consumed so that the employees complete the task requested, minimizing the cost, maximizing service quality and observing all the restrictions that may appear in the context*. Several micro-objectives have been deduced from this macro one, including:

i. Generate itineraries without delays or idle time between two different tasks.

ii. Generate itineraries that minimize the cost associated to the length of the task.

iii. Optimize the working time of the employees to avoid paying overtime or that they work less time that the legal number of hours per week.

iv. Minimize the cost of the employees.

v. Minimize the cost of the additional resources.

vi. Optimize the quality of service.

vii. Balance the number of tasks performed by different resources.

These micro-objectives are, of course, in relation with the entities in the ontology in Figure 7-18. For the two fitness functions detailed below, we will denote by *R* the set of resources, by *PE* the set of *PlannedElement*, and to each vehicle R_i , we associate the ordered list $P_i = [p^i_1 , ..., p^i_n]$, $p^i_j \in PE$ of *PlannedElement* assigned to the resource. Now the objectives can be formally defined as below (which will form the C_{opt} function of the plan model):

1. *Minimizing the cost associated with certain point in delay or in advance*: Each activity between two *PlannedElements* p_j and p_{j+1} is represented in a temporal with three values, *RequestedDate* (RD), *Duration*(D) and the estimated time to arrive to the next task (*TFN*) i.e. overhead caused by the resources being consumed.. The duration estimates the time needed in a point to complete the activity. Now, the time in minutes it takes for a resource to complete a task *i* and move to task *j* in time *t* be represented as TFN_{ijt} (because of the three dimensional nature of the entity). The cost is represented through a piecewise function:$f_0(x)$, where $x = RD_{j+1} - (RD_j + D_j + TFN_j)$. If this difference is negative (task is delayed), the cost is quadratic; otherwise (task is on time), the cost is linear. Hence,

$$f_0(x), = \begin{cases} x^2, & x < 0 \\ x, & x \geq 0 \end{cases}$$

Therefore, the objective can be formalized as:

$$\min \sum_{v \in V} \sum_{p \in P_i} f_0(RD_{j+1} - (RD_j + D_i + TFN(p_j, p_{j+1}, RD_{j+1})))$$

2. *Minimizing the cost associated to the resource consumption and associated overhead cost*: Ideally, the resources should be assigned to the tasks which are at close proximity in both location and dates. Considering TFN_{ijt}. as the associated overhead cost on any resource R_j . Therefore, if $x = (p_j , p_{j+1}, t)$ where *t* is the time of the day when the task needs to be done, this objective can be formalized as:

$$min \sum_{v \in V} \sum_{p \in P_i} costTask(v_i, TFN(p_j, p_{j+1}, RD_{j+1}))$$

7.3.2.3. EVOLUTIONARY OPERATORS

In the process of applying genetic algorithm, it's necessary to define the structure of chromosome and the evolutionary operators. In the problem domain being addressed in this scenario, when the GA finishes its execution, it returns a planning based on the ontology, to which the structure of the chromosome is adapted as depicted in Figure 7-19. For the set of all the resources, the chromosome associates then a person and a list of points (*PlannedElements*) that the resource and person needs to attend (this list of *PlannedElements* is the *PlanTimeLine* associated to each resource). All list of *PlannedElements* are sorted by ascending operation date (with the object property operationDate(p) in the ontology).

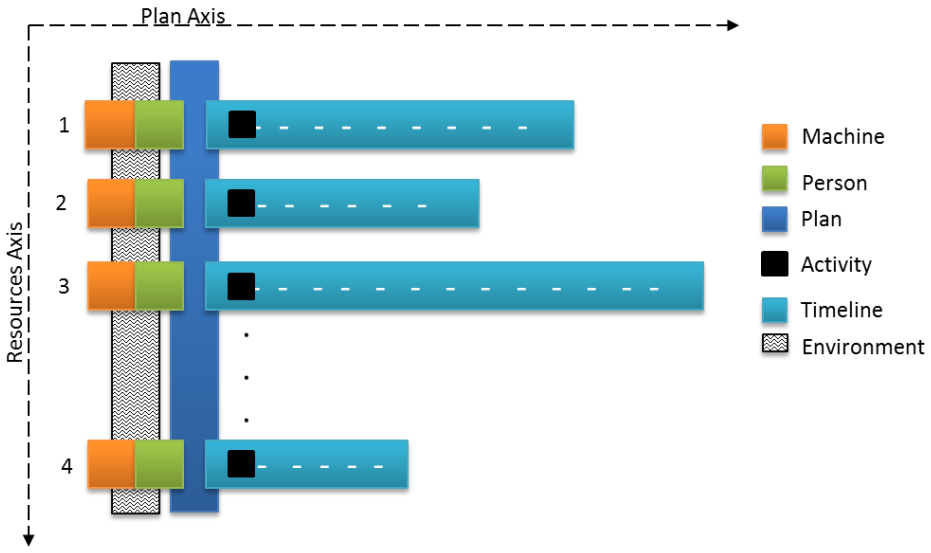


Figure 7-19 Structure of the chromosome

Now, for the population to evolve, it is necessary to define a set of evolution operators by taking into account all the aspects that are necessary for the planning to tend towards a possible final solution. There are two major axes that differ in the structure of a chromosome, the distribution of the resources and the *PlannedElements* on the set of resources. Making evolutionary changes in the two axes can affect all costs associated with the objectives, generating new populations with individuals of better quality. The operators to be defined along both the axes are crossover and mutation.

PlannedElementCrossover: As it's clear from the ontology presented in the plan model, *PlannedElements* may be regrouped in a *Task*. Each *PlannedElement* belongs to at most one *Task*. We denote by *inT(p)* the set containing *p* and all the *PlannedElements* in the same *Task* as *p*. In Figure 7-20, the big long top rectangle represents the list of all the *PlannedElements* sorted by ascending time of operation.

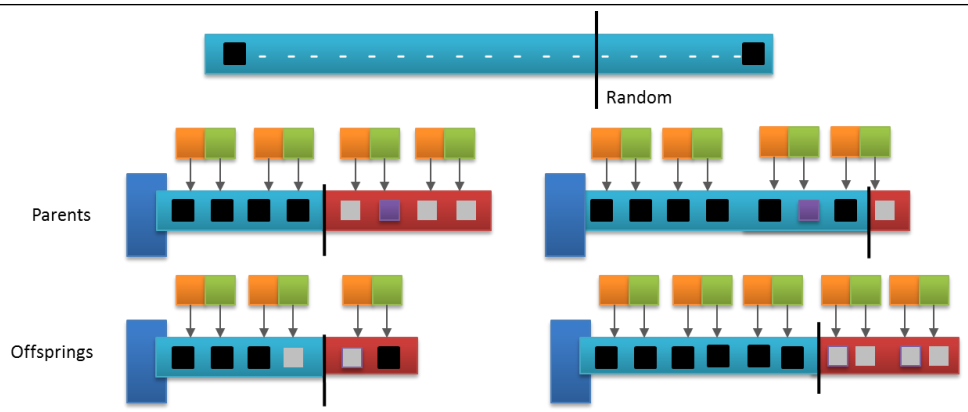


Figure 7-20 Crossover Operator for PlannedElement

In order to have crossover between two individuals, named parents in the figure, a *PlannedElement* p_r is randomly chosen in this list. For each *PlanningTimeLine i*, we have the list P_f^i (resp. P_m^i) of *PlannedElements* affected to resource *i* in the *father* (resp. *mother*) individual. The set P_c^i of *PlannedElement* affected to resource *i* in the new offspring individual, which is defined as:

$$P_s^i = \{p^f \in P_f^i;\ \exists p \in inT(p)\ RD(p) <= RD(p_r)\}\ \cup \{p^m \in P_m^i;\ \forall p \in inT(p)\ RD(p) > RD(p_r)\}$$

Informally speaking, the offspring has the assignation of its father for early *PlannedElements*, and of its mother for later ones. Or we can define another offspring by reversing the role of the father and mother. The corresponding operations are schematically shown for one *PlanningTimeLine* at the bottom of Figure 7-20.

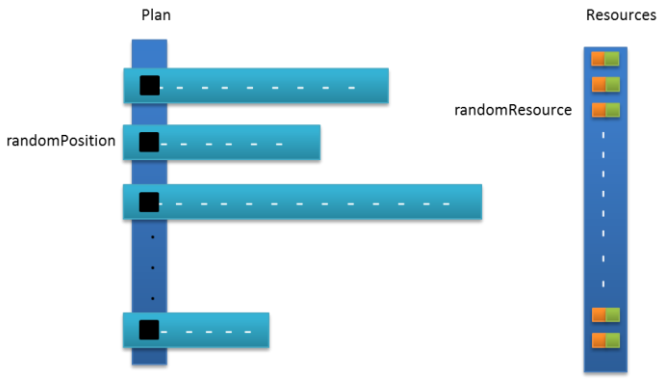


Figure 7-21 Resources mutation operator

ResourceMutation: This operator is implemented by taking a random resource of the list of all possible resources that are assigned to some task in the plan. Then we make a simple swap between the selected random resource and a randomly selected resource in the individual, only if the number of persons is the same in both tasks to which the resource is allocated. This simple mutation operator is as shown in Figure 7-21.

7.3.3. *TECHNOLOGY IMPLEMENTATION*

In the process of realization of the technical solution for this scenario, the technical solutions that have been developed in scenario 1 and scenario 3 are stacked together. Scenario 1 forms the sensing layer while the behavioural layer of scenario 2 is adopted to model the tasks and activities in the scope of construction project planning, as depicted in Figure 7-22. The behavioural layer is enhanced with the plan model that has been enhanced for the purpose of this scenario.

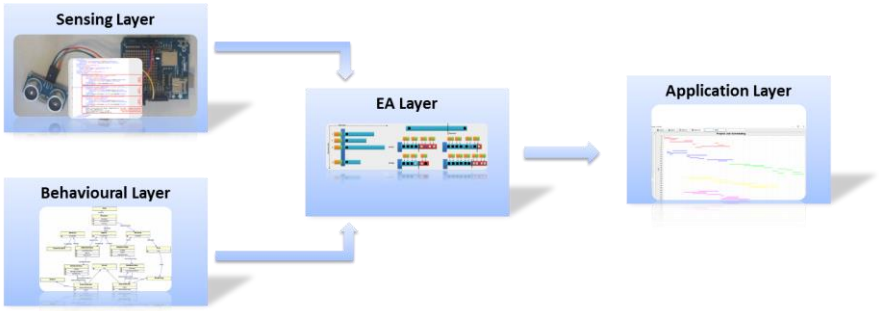


Figure 7-22 Layers of technical solution of scenario 3

The EA layer implements the chromosomes representation and operations as described in the previous sub-section. The EA was programmed using JAVA and run using Eclipse interface on a personal PC. The implementation is based on the results of the implementations techniques developed in two open source projects EASEA[27] and OptaPlanner[28]. The generic algorithm that has been implemented in the scope of this solution is as explained in Appendix A section 11.2. The main classes that have implemented are as depicted in Figure 7-23. Note that the way the EA algorithm works is dependent on the detection of new situations from real-time data. The detection of new situation affects the evaluation of fitness functions and the generation of genotypes hence initial populations and the new generations generated through genetic operators.

In the implementation that has been done to experiment with the formulated EA model based on the real world problem of multi-variant function; the functionalities have been developed for: setting up a fitness function, choosing a representation for the problem and fine-tuning the parameters. This implementation is modular and provides extendable solution to implement different types of genetic operators, selection methods and optimization algorithms.

[27] https://github.com/EASEA/easea

[28] http://www.optaplanner.org/code/sourceCode.html

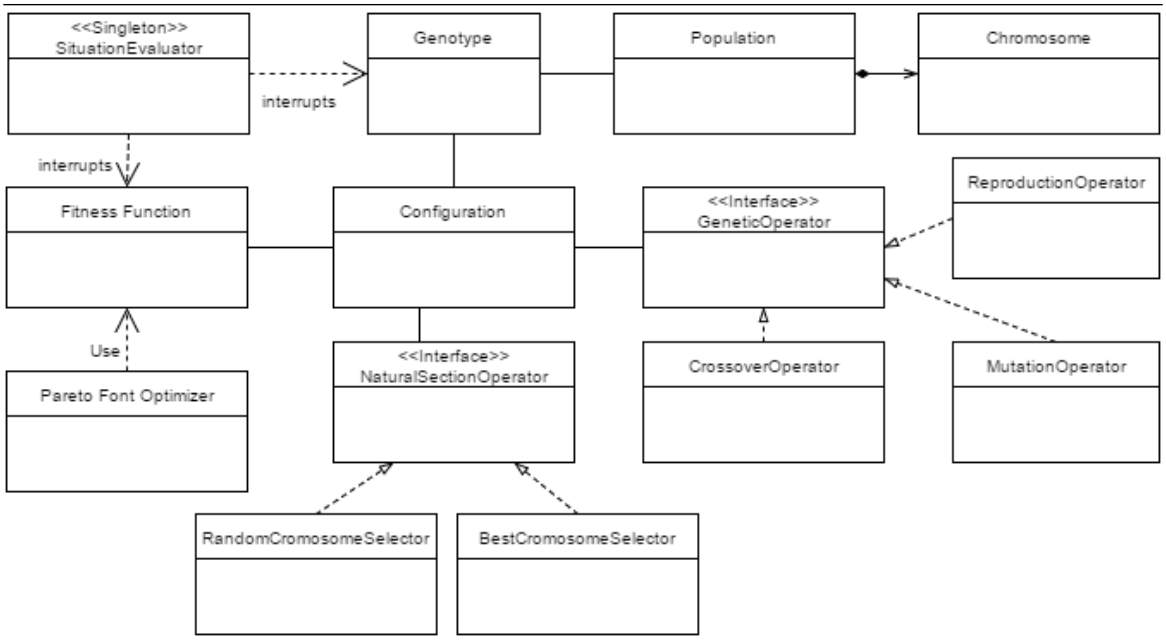


Figure 7-23 Class diagram of implementation of EA

During the course of implementation and experimentation we have observed that GAs are expensive in terms of resource consumption and time it takes for the completion of evolutionary cycle. The cost comes from the total memory footprint (*genomeSize* × *populationSize*) which is often very large for real projects. And the convergence time to an acceptable solution is too long, often leading the computers towards sudden crash. So, considering the complexity of the problem it's necessary to utilize the massively parallel implementation as proposed also by the EASEA platform. It is important to note that in the scope of this implementation we haven't considered the parallel implementation but reduced the problem domain to simpler case to have experimental results.

7.3.4. *OBSERVATIONS*

The implementation has been tested with smaller subset of the original concreting plan and with necessary modifications to fit the EA optimization model. For the evaluation of the algorithms we have considered the optimization functions as descried in section 7.3.2.1. The input data of the algorithm consists of a number of projects and corresponding tasks. As mentioned in section 7.3.2.1, each task includes constrains regarding the time, the cost, the number of resources, etc. All the three scenarios are evaluated using a population of 500 individuals during 50 generations. The probabilities of the genetic operators are 0.8 for the crossover and 0.1 for the mutation.

Figure 7-24 shows the initial state of the project plan, consisting of three projects and each project has 12 tasks. Each task has the start and the end date and each task is associated with some resource (not shown in the figure though). In the beginning, the plan is made without having overlap between the tasks start and end date. Figure 7-25 shows the ongoing optimization operation. Here, we can observe how the algorithm assigns the

start and end date for each of the operations. Figure 7-26 shows the final state where the tasks of the projects are optimized based on the resources that they require and without hindering the operation of the other tasks.

Figure 7-24 Initial State

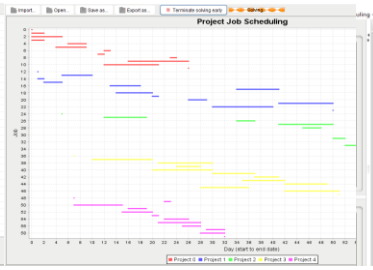


Figure 7-25 State after 20 generations

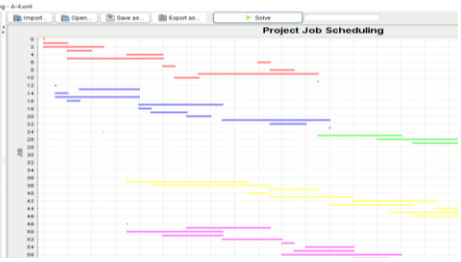


Figure 7-26 Final state

As it has been observed in this preliminary experiment, we could achieve optimized *PlanTimeLine*, for each *PlannedElement.* But, experimenting this in the real industrial scenario might need some improvements to handle the effect of real-time delays caused by human errors or machine malfunctioning. So, utilizing this algorithm in the real CPS scenario is still a future work.

SECTION IV: DISCUSSIONS AND PROSPECTIVE

In this section of the dissertation is presented the discussion of the research results with clear indication of the research results in relation to the problem domain and research questions identified at the beginning of the dissertation. This section is divided into two chapters: ***Chapter 10: Research Results and Scientific Contributions and Chapter 11: Conclusion and Future Work***

Chapter 10: *This chapter is dedicated towards making the synthesis of the research results providing an explanation of the research track from the problem to the solution. It also provides the rational of the research results in respective to the research questions and research gaps identified in Section I. At the same time this chapter presents the integration of the research work with various industrial projects and fellow researchers from other domain to have cross domain research experience and the results achieved through research collaboration. The chapter ends by presenting the list of publications that have been made during the timeframe of the research work.*

Chapter 11: *This chapter provides the concluding remarks on the research work and also presents the insights into the future works.*

On the whole by the end of this section, it is expected that the reader will have clear understanding of the research results of this PhD work and have the light into the future research and technical directions that can be undertaken towards achieving smart CPSs .

8. RESEARCH RESULTS AND SCIENTIFIC CONTRIBUTION

To raise new questions, new possibilities, to regard old problems from a new angle, require creative imagination and marks real advance in science.

- Albert Einstein

This chapter is dedicated towards making the synthesis of the research results providing an explanation of the research track from the problem to the solution. It also provides the rational of the research results in respective to the research questions and research gaps identified in Section I. At the same time this chapter presents the integration of the research work with various industrial projects and fellow researchers from other domain to have cross domain research experience and the results achieved through research collaboration. The chapter ends by presenting the list of publications that have been made during the timeframe of the research work.

This chapter provides discussion on the PhD research work and the analysis of the results. Results are analysed from two important aspects i.e. from problems to results and from research gaps/aimed contributions to results. Let's start with the discussion on the rational on result results- which mainly focuses on discussion of this research work life cycle starting with 'identification of problem' and ending with 'exploitable results'.

8.1. RATIONAL ON RESEARCH RESULTS

The research work that has been undertaken was ambitious considering the diverse domains of science connected with the research problem. The research work different domains of science and engineering such as biology, theory of computation and software engineering. In order to build bridges between concepts from such confluence of research areas, section 1.4 provides a brief discussion including the identified gaps and the aimed contributions that can be made in each area. In chapters 3 and 4, such diverse research areas are studied in depth to provide a solid understanding of related areas and discussion on past and ongoing research works. Existing scientific knowledge on theory of evolution forms the basis of this research work. At the same time works of Alan Turing in 1940's provides the solid theoretical background for this PhD research. Whereas, recent developments in sensor networks, dynamic networks, complex systems and software engineering provide the base for the technological solution formulated in this research work. Evolution has been well studied in biology which has basically been inspired by the fact that the environment of the living beings is very different at different periods of time and so are the functional features of the living organism in corresponding periods. The study of such organisms manifests astonishing variation as a consequence of genetic recombination and random genomic changes. Therefore biological variation provides the "raw material" for evolutionary change. Theory of computing provides the basis for computational model and has been long studied for different computational models, but has been less explored in the CPS domain which have hybrid behaviour i.e. incorporation of both physical and computational processes. At the same time other areas presented in

literature review, are necessary to establish scientific and technological base for the development of the proposed theoretical and technical framework for evolutionary CPS. Theory of evolution have been used as the corollary for the system evolution i.e. system changes based on changes in business and/or technical requirements. In most of the previous work system evolution has been studied under controlled environment i.e. system developer is feed with the changes and system is developed based on the new requirements. But, this research work aims to take it one step further by integrating system with intrinsic requirements changing environment, which is a necessity for CPS.

From an industrial application point of view, this research work is particularly interesting from smart factory perspective, which aims to have modular structured factories in which CPSs monitor physical processes, create a virtual copy of the physical world and make decentralized decisions [288]. The next generation network provides necessary communication infrastructure where CPSs communicate and cooperate with each other and with humans in real time, and via the emerging model of Internet of Services. This allows the creation of new business value chains via integration of cross organizational services. In this scenario, study of evolutionary systems in close relation with CPSs paved an important understanding of sensorial and behavioural modelling necessary for evolutionary CPS along with necessary technological foundation and solution modules for interfacing with environment, situational awareness and processes optimization.

8.1.1. *FROM PROBLEM TO SOLUTION*

The research undertaken in this PhD work has gone through a number of cycles between theoretical and technical research. But it all started with identification of problem related to systems that are similar to living organism – from the perspective that both have to deal with dynamic nature of working environment. So, the basic problem or the confusion on the head of the author was if it's possible to achieve systems that are capable of finding solutions under dynamic constraints. The obvious choice to study such types of systems and environment was CPS. This quest for knowledge inspired the formulation of research question- on the possibility of integrating learning and automated reasoning in the design and implementation processes of CPS, to enable evolutionary behaviours. The main research question has been divided into sub-questions such that each question tries to find solution for detection of the problem, response to the problem and impact of the solution (c.f. section 2.1). These sub-questions are important ones to be answered for further enhancement of the state of art of CPS. In the CPS domain the engineering problems faced daily is managing dynamics, time, and concurrency in heterogeneous (interconnected) systems where the amount and complexity of intelligence (the cyber part) is growing rapidly and where software implementations are a major portion of system design, validation and ultimately verification. The research questions have captured this theme paving a clear path for implementation (SQ1 and SQ2) and validation/assessment (SQ3). Figure 8-1 shows the general flow of this PhD work that

started with a problem and ended with exploitable solutions. The next step was the formulation of hypothesis, the main highlight of which is development of learning mechanism over the behavioural model of the system. If the behavioural model of the CPS can be formally modelled and represented along with the control mechanism based on events and actions, then systems adaptive loops can be used for learning to derive evolutionary paths.

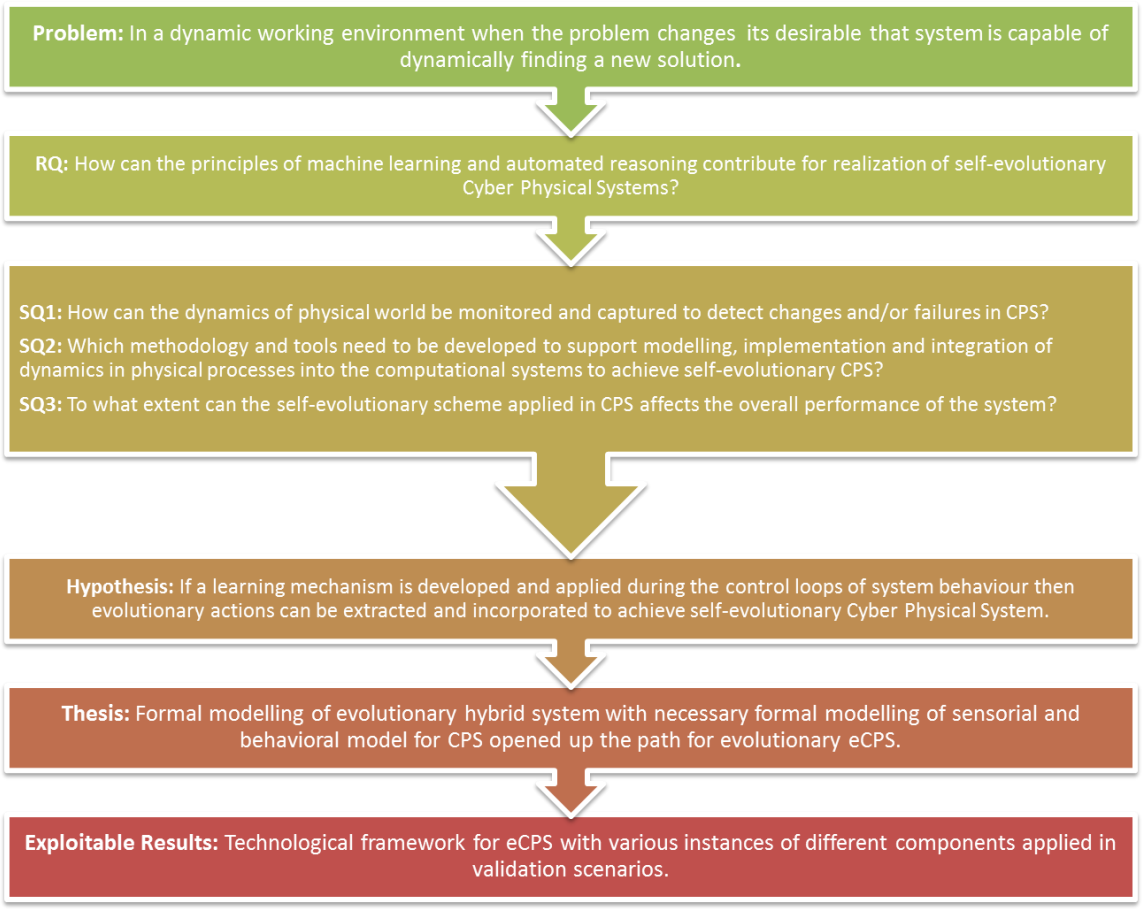


Figure 8-1 Flow of research work- from Problem to Results

The hypothesis has mainly lead towards the formulation of theoretical foundation for eCPS. On the whole, the major outcome of this PhD research is this dissertation, which provides the thesis which can be summarized as "***Formal modelling of evolutionary hybrid system with necessary formal modelling of sensorial and behavioural model for CPS opened up the path for eCPS.***" For the detailed understanding of this statement follow chapter 6. The thesis is supported with necessary technological foundation which is discussed in chapter 0. Each of the results of this PhD work will be further presented in detail in the following sections. But to make a quick summary, this research work provided exploitable technical solutions, which have been integrated and tested in industrial scenarios.

Table 8-1 provides a summary of the research results in relation with the research questions.

Table 8-1 Research results in relation with research questions

ID	Question	Related development and outcomes
SQ1	How can the dynamics of physical world be monitored and captured to detect changes and/or failures in CPS?	A. Behavioural modelling of CPS (c.f. section 6.1.3) B. Sensorial modelling of CPS (c.f. section 6.1.4) C. Technological foundation for CPS (c.f. section 6.2.2) D. Technological solution on Global Sensorial Network (c.f. section 7.1) From the results i.e. A, B we have presented a way to represent dynamics in CPS utilizing the behavioural and sensorial models; C, D provides corresponding technological solution.
SQ2	Which methodology and tools need to be developed to support modelling, implementation and integration of dynamics in physical processes into the computational systems to achieve self-evolutionary CPS?	A. Formal model of evolutionary machines and CPS (c.f. sections 6.1.1 and 6.1.1.1 and 6.1.2) B. Methodology for design, implementation and integration of (e)CPS (c.f. section 6.2) Results A and B provides the methodology for modelling and implementing evolutionary CPS, which has been adopted in implementing the validation scenarios.
SQ3	To what extent can the self-evolutionary scheme applied in CPS affects the overall performance of the system?	A. Dynamic deployment framework (c.f. section 6.2.2.3) B. Plan optimization with evolutionary algorithms (c.f. section 7.3) Results A and B provides the methodology that can allow the implementation of self-evolutionary scheme. But, this was not fully experimented with industrial use cases to have complete and certain answer to SQ3 and remains as an interesting future work.

8.1.2. *DEVELOPMENTS ACHIEVED TOWARDS FORESEEN CONTRIBUTIONS*

In section 1.4 we have provided a discussion on most prominent research areas in the scope of this dissertation, which were Evolutionary Computation, Knowledge Engineering, Sensing Systems and cyber Physical Systems. Even though the research

work touched many more other research domains such as software engineering, control theory, theory of computation, formal methods etc. the before mentioned areas remained the top level areas based on which achievements can be discussed. Table 8-2 presents the developments achieved towards the foreseen contributions as presented in section 1.4.

Table 8-2 Aimed Contributions vs Achievements in identified key research domains

Domain	Aimed Contribution	Achievements
Evolutionary computation	In this area the aimed contribution is thus the knowledge injection methodology in evolutionary algorithms. It will in over-all be a step ahead to overcome the short coming of evolutionary algorithms due to the lack of domain knowledge (i.e. exponentially increasing search space), so that we can reduce the time complexity of the evolutionary algorithms. At the same time it is expected to work in the methodology for enhancing knowledge base based in the results of evolutionary computation, so that both the knowledge base and the evolutionary algorithm improve over time.	In this regard, the major contribution is the development of methodology for knowledge injection in the EAs as explained in section 6.1.5 and corresponding implementation as described in section 7.3, which has been tested with sub-set of concreting plan as observed in the construction industry.
Knowledge Engineering	A specific area for contribution is application of generic programming for knowledge extraction tasks. It is thus necessary to address a proper knowledge representation and reasoning methodology over real-time environmental data. The knowledge model should also consider emergence of relevant correlations while maintaining consistency of the overall knowledgebase. Thus, in general the expected contribution in this domain is novel methodology for knowledge engineering in real-time systems by considering the capability to model events, situations, context and reason over them, while maintaining temporal constraints.	This gap has been addressed mostly by the theoretical foundations described in section 6.1, which provides conceptual model for representing different factors in the CPS domain. In addition to the reference models, a number of knowledge representation models have been presented in scenarios described in sections 7.2 and 7.3 tested with industrial use-cases.

Sensing Systems	Sensing from the physical world is an important aspect for this research work. Thus, the first important expected contribution is the development of generic solution for seamless integration of new devices into the existing system. The second contribution will be the solution for virtualization of physical devices. And the final expected contribution is towards creating algorithms that can predict and estimate missing values in a stream collected from physical/virtual sensor to help in predictive monitoring and analytics.	This gap has been addressed exclusively by the theoretical foundation on sensorial modelling as described in section which provided methodology to effectively represent sensor, virtual sensors, actuators and reinforcement sensing model in sensorial network. The research work also presents technical implementation for sensing framework as described in section 7.1.
Cyber Physical Systems	Methodology for modelling CPS is one important area that this research work will explore into aiming to provide considerable contribution. Work in modelling formalism to properly express both the cyber and physical domain of CPS is thus an expected contribution. The other foreseen contribution is building up technological framework that can be used for realization of scalable and robust CPS, along with the implementation of necessary generic modules necessary for building CPSs.	This research gap has been addressed not only by the theoretical model as described in section 6.1.6, but also by the design and implementation methodology presented in section 6.2. This design methodology has been followed in implementing the scenarios as descried in chapter 7, leading towards some interesting results.

8.2. INTEGRATION WITH OTHER RESEARCH ACTIVITIES

During the period of PhD research, author collaborated with different fellow researchers at *Faculdade de Ciências e Tecnologia - Universidade Nova de Lisboa (FCT-UNL)*; *Group for Research in Interoperability of Systems- Centre of Technology and Systems (GRIS-CTS-UNINOVA)* and various research projects funded by *European Union (EU)* and *Quadro de Referência Estratégico Nacional (QREN)*. Following subsection provides the discussion on research integration and important results

8.2.1. *INTEGRATION WITH FELLOW RESEARCHERS*

Author collaborated with fellow candidates at Department of Electrical Engineering, FCT-UNL pursuing PhD in Electrical and Computer Engineering. Collaborative research

not only provided inputs for this dissertation, but also could collaboration in other two PhD dissertation viz. "*Knowledge Management Framework based on Brain Models and Human Physiology*"[289] and "*A new MDA-SOA based Framework for Intercloud Interoperability*" [290]. In the course of these collaborations, the major contribution by the author was more on the technical solutions formulation and implementation, which has been useful in the course of the author's research work too. Besides the collaboration has lead towards two important publications:

- Luis-Ferreira, S. Ghimire, and R. Jardim-gonçalves, "*Framework for Knowledge Management Towards Human Centric Internet of Things and Sentiment Analysis*," in *ASME 2014*, 2014. [238]
- T. Nodehi, S. Ghimire, R. Jardim-Goncalves, and A. Grilo, "*On MDA-SOA based Intercloud Interoperability framework*," *Journal Computational Methods in Social Sciences*, vol. 1, no. 1, 2013. [291]

Besides these publications, author has collaborated with the same first authors in some other publications which are presented in section 8.3.

8.2.2. *INTEGRATION WITH INDUSTRIAL PROJECTS*

The research work was started under the scope of QREN – COMPETE project VortalWAY - Cloud Computing for Electronic Platforms (http://vortalway.vortal.pt/), the FITMAN project EC 7th Framework Programme, Future Internet Technologies for MANufacturing industries, under grant agreement n° FITMAN 604674 (http://www.fitman-fi.eu) and the C2NET project EC HORIZON2020 Program under grant agreement n° C2NET 636909 (http://www.c2net-project.eu/).

8.2.2.1. VORTALWAY

The VortalWay project's main goal was to develop upon a B2B platform a set of procurement services based on the Cloud Computing service approach, which was applied in e-procurement business scenarios on wide range of business sectors such as Building & Construction, Public Administration, Health, etc. The main contribution that was made by the project was platform to bridge interoperability among B2B platforms, namely the ones related to procurement activities. VortalWay project, also delivered new advanced features for e-catalogues, through data categorization based on a sophisticated product mapping mechanisms and semantic search within different e-catalogues, enhancing with this, user requests for certain items. The major results from the project were:

- [Publication] S. Ghimire, R. Jardim-Goncalves, A. Grilo, and M. Beca, "Framework for inter-operative e-Procurement marketplace," in *Proceedings of the 2013 IEEE 17th International Conference on Computer Supported Cooperative Work in Design, CSCWD*

2013, 2013, pp. 459–464. [292]

- [Publication] S. Ghimire, R. Jardim-Goncalves, and A. Grilo, "Framework for catalogues matching in procurement e-marketplaces," *Inf. Syst. Technol. (CISTI), 8th Iber. Conf.*, pp. 1–6, 2013. [293]
- [Technological Solution] Portal for inter-operative e-procurement marketplace. [private deployment by the company Vortal, technical description in publication [292]]
- [Technological Solution] Implementation of matching and learning algorithm in the domain of Request for Quotation (RFQ) and product catalogues. . [private deployment by the company Vortal, technical description in publication [293]]

On the whole the major contribution of this project towards this disseration is understanding of ***interoperability science, cloud technologies, service oriented architectures and learning algorithms*** and has provided inputs in the the formulation of the technical frameworks presented in ***chapter 0*** , specifically in the top most layer for continious integration and dynamic deployment

8.2.2.2. FITMAN

FITMAN aimed to develop the Factories of the Future and business in Europe by using the Fi-Ware core platform of services. FITMAN main results were reference architecture, based on Future Internet technologies, for each of the three manufacturing domains identified by the EFFRA (European Factory of the Future Research Association)[29] i.e. digital factory, smart factory and virtual factory. Solutions on smart factory domain focused on agile manufacturing, process automation control, and tools for sustainable manufacturing. While, digital factory domain is focused on improving the design of production and manufacturing systems and product life cycle management. Virtual factory addresses supply chain management, product-service linkage and management of distributed manufacturing.

The FI-WARE platform provides services for European companies to perform wide range of IT operations such as solutions for cloud services, IoT, complex event processing, data visualization, 3D modelling etc. Within the scope of FITMAN project existing services are used to create new values in the existing business processes in manufacturing industries domain. The solutions used in the technological solution of this PhD research work during instantiations of the proposed Framework. Major outcomes in the scope of this project that contributed this PhD work are:

- [Publication] F. Luis-Ferreira, S. Ghimire, and R. Jardim-Goncalves, "Brain inspired health monitoring supported by the cloud," in *IFIP Advances in Information and Communication Technology*, 2015. [266]

[29] http://www.effra.eu/

- [Publication] S. Ghimire, F. Luis-Ferreira, T. Nodehi, and R. Jardim-Goncalves, "IoT based situational awareness framework for real-time project management," *International Journal of Computer Integrated Manufacturing,*2015. [294]
- [Technological Solution] Portal for collaborative project planning and real-time information tracking for construction industry. Industrial partner-Consulgal. [application is deployed at http://fitman.uninova.pt/, core technological description in the publication [294]].
- [Technological Solution] Shop floor data collection specific enabler – provides technological solution for efficient data collection from the manufacturing shop floor by using technologies like RFIDs and sensor networks. [No public deployment, source and virtual machines are available from Fitman Innovation lab (FML) portal. Core technological description is provided in Fitman Catalogue and also in the publications [294] and [295].

On the whole the major contribution of this project towards this disseration is understanding of ***cloud technologies, service oriented architectures, Internet of Things, Situational Awareness and complex event processing*** and contributed towards the implementaiton of ***scenario 1*** and ***scenario 2*** as discussed in Chapter 7.

8.2.2.3. C2NET

C2NET aims to build a novel Cloud Architecture to provide ubiquitous tools supporting collaboration among value chain partners and providing advanced algorithms to achieve holistic global and local optimization of manufacturing assets and to respond faster and more efficiently to unforeseen changes. The main goal is to support optimization collaborative demand, production and delivery plans, by mastering master complexity of the supply network. C2NET toolset will enable the rendering of the complete set of supply chain management information on the any digital mobile device (PC, tablets, smartphones) of decision makers enabling them to monitor, visualize, control, share and collaborate. This is an ongoing project, where the author has been involved in the later phase of the PhD research. The major outcomes are:

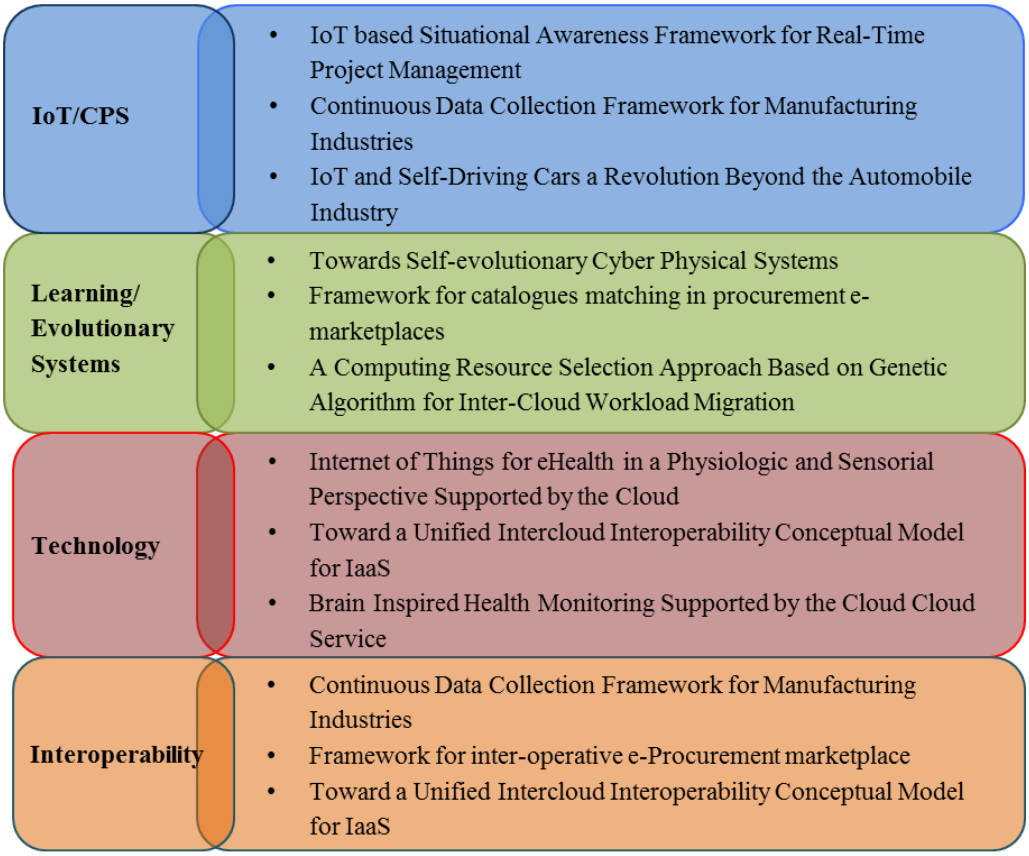
- [Publication] S. Ghimire, R. Melo, J. Ferreira, and C. Agostinho, "Continuous Data Collection Framework for Manufacturing Industries," in *On the Move to Meaningful Internet Systems: OTM 2015 Workshops*, vol. 2, 2015 [267]
- [Technical solution]: C2NET collaboration ontology- Knowledge Base for capturing the collaborative situations in manufacturing industries.[ongoing work]
- [Technical solution]: Data collection Framework – framework for data collection from manufacturing shop-floor by considering almost real-time constraint and the heterogeneous nature of data source [ongoing work, technical description in paper [267]]

On the whole the major contribution of this project towards this disseration is understanding of ***Internet of Things, Real-time systems, CPS, protocol adaptation and ontologies;*** and contributed towards the implementaiton of technical solution in ***scenario 1***as discussed in Chapter 7.

8.3. PUBLICATIONS

In the course of the PhD research publication of research results is an important activity (also marked in research methodology c.f. chapter 0). Results from the research work has been published in conference papers or journals addressing several areas that that been in the scope of the work. Table 8-3 presents the papers related to those areas are listed according to their fit to the same areas.

Table 8-3 Publications within related research area



Additionally, Table 8-4 provides all the list of the publications that have been made during the time frame of this research work, ordered by year of publication.

Table 8-4 List of Publications

1	**Sudeep Ghimire**; Luis-Ferreira F.; Nodehi, T. and Jardim-Goncalves, R.: "*IoT based Situational Awareness Framework for Real-Time Project Management*". International Journal of Computer Integrated Manufacturing	2016

2	Fernando Luis-Ferreira; **Sudeep Ghimire**; João Sarraipa and Ricardo Jardim-Gonçalves : *"IoT and Self-Driving Cars a Revolution Beyond the Automobile Industry*", IMEC 2016	2016
3	**Sudeep Ghimire**; Raquel Melo; José Ferreira; Carlos Agostinho and Ricardo Goncalves: "*Continuous Data Collection Framework for Manufacturing Industries*". OTM Workshops 2015: 29-40	2015
4	Fernando Luis-Ferreira; **Sudeep Ghimire** and Ricardo Jardim-Gonçalves: "*Brain Inspired Health Monitoring Supported by the Cloud*". DoCEIS 2015: 273-281	2015
5	**Sudeep Ghimire**; Fernando Luis-Ferreira; Ricardo Jardim-Gonçalves and Tahereh Nodehi: "*Towards Self-evolutionary Cyber Physical Systems*". ISPE CE 2014: 547-554	2014
6	Tahereh Nodehi; **Sudeep Ghimire** and Ricardo Jardim-Gonçalves: "*A Computing Resource Selection Approach Based on Genetic Algorithm for Inter-Cloud Workload Migration*". ISPE CE 2014: 271-277	2014
7	Fernando Luis-Ferreira; **Sudeep Ghimire** and Ricardo Jardim-Gonçalves: "*Internet of Things for eHealth in a Physiologic and Sensorial Perspective Supported by the Cloud*". ISPE CE 2014: 790-795	2014
8	Luis-Ferreira, Fernando; **Ghimire, Sudeep**; Zdravkovic, Milan; Jardim-Goncalves, Ricardo: "*Framework for knowledge management towards human centric internet of things and sentiment analysis*". IMECE 2014	2014
9	Tahereh Nodehi; **Sudeep Ghimire** and Ricardo Jardim-Gonçalves: "*Toward a Unified Intercloud Interoperability Conceptual Model for IaaS Cloud Service*". MODELSWARD 2014: 673-681	2014
10	T. Nodehi; **S. Ghimire**; R. Jardim-Goncalves, and A. Grilo: "*On MDA-SOA based Intercloud Interoperability framework*," Journal of Computational Methods in Social Sciences, vol. 1, no. 1, 2013	2013
11	**Sudeep Ghimire**; Ricardo Jardim-Gonçalves; António Grilo and Miguel Ferro de Beca: "*Framework for inter-operative e-Procurement marketplace*". CSCWD 2013: 459-464	2013
12	**Sudeep Ghimire**; Jardim-Gonçalves, R. and Grilo, A.: "*Framework for catalogues matching in procurement e-marketplaces*". In: Information Systems and Technologies (CISTI), 2013 8th Iberian Conference on Information Systems and Technologies.	2013
13	Grilo, A; Jardim-Goncalves, R and **Ghimire, S**.: "*Cloud-Marketplace : New Paradigm for e-Marketplaces*". PICMET'13., p 555–561.	2013

9. CONCLUSION AND FUTURE WORK

Now this is not the end. It is not even the beginning of the end. But it is, perhaps, the end of the beginning.

- Winston Churchill

This chapter presents the concluding remarks on the research work and the results that have been achieved so far. It also presents the light into the future research and technical activates that can be undertaken in the domain of CPS. By the end of this chapter, it is expected that the reader will have clear understanding of the research results of this PhD work and motivations for the future research and technical directions that can be undertaken towards achieving smart CPSs.

9.1. CONCLUSION

As state in the beginning of this dissertation (c.f. sections 1.2 and 1.3), this research has been a pursuit for studying and understanding the phenomena of evolution in the scope of cyber physical systems. This research work has been a step towards new generation of cyber physical systems i.e. self-evolutionary cyber physical system (eCPS). One of the main visions is formulating the methodology for realization of self-evolutionary cyber physical systems (eCPS). In the course of realization of eCPS, it is necessary to enhance the models used for representation of components, objects and events etc. with sensorial data as collected from environment and formulating methodology for reasoning over such data by considering contextual data, more importantly temporal dynamics.

The initial research challenge was to find interesting and useful lessons on how to design, model and develop computing systems that will be deployed in highly dynamic and uncertain environment. For that purpose, one obvious aspect to be studied is the methodology to capture real-time data and extract useful information such as events and situations, so that the system can take actions based on such observations. The reactions to such observed situations can lead towards the accumulation of knowledge or experience of the nature of the system's environment and eventually lead towards evolution on functional behaviours. In the course of this dissertation, "*Theory of Natural Evolution*" has been an important reference, not only because it has a long history and acceptance among researchers around the world, but especially it gives insight on how living organism evolve and the impact of environment on evolution. Characteristics of environment are common factor for biological systems and cyber physical systems (even though CPS, can be deployed in more controlled environment).

As depicted in Figure 9-1, this dissertation has lead towards understanding the evolutionary behaviour in cyber physical systems by considering the intrinsic features of real-world environment viz. dynamicity, uncertainty and time. In the computing environment specific studies have been made to understanding evolutionary computational model. At the same time important aspects for modelling eCPS i.e. domain knowledge, functional modelling, behavioural modelling and goal/preference based

computing has been studied. This study has thus lead towards forming the foundations for further research in eCPS.

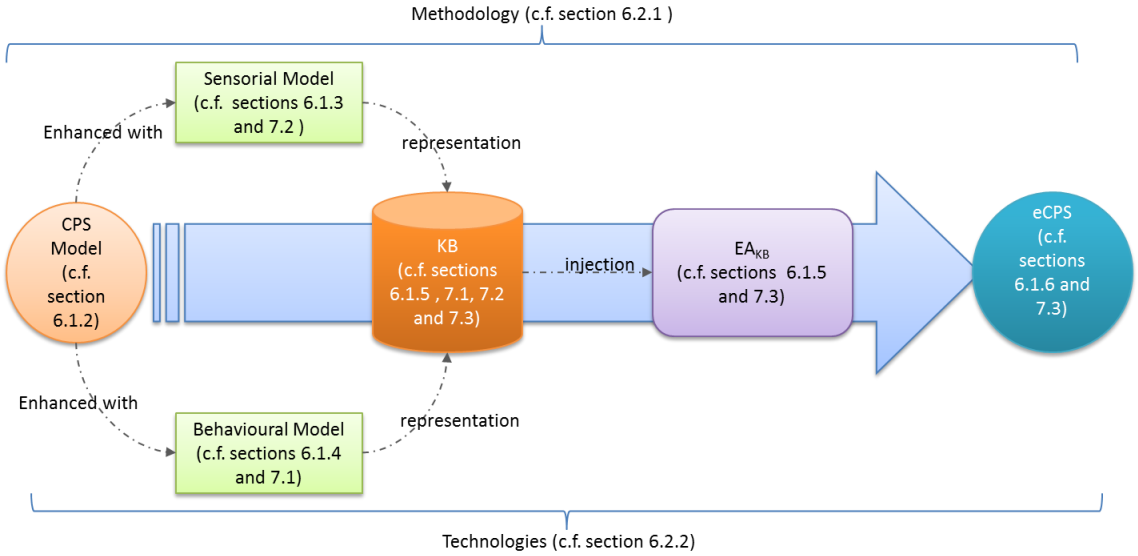


Figure 9-1 From CPS towards eCPS and core contributions in the dissertation

One of the most important outcomes of this desertion is the formulation of formalism that can be used for modelling evolutionary cyber physical systems along with formulating the methodology for realization of self-evolutionary cyber physical systems (eCPS). During this research activity it has been clear that, it is necessary to enhance the models used for representation of components, objects and events etc. with sensorial data as collected from environment and formulating methodology for reasoning over such data by considering contextual data, more importantly temporal dynamics. The sensorial and behavioural modelling formalism (c.f. sections 6.1.3 and 6.1.4) and corresponding technological implementations (c.f. sections 7.1 and 7.2) have not only validated the above mentioned point but also provided technological foundation for realizing such systems. Situational modelling and corresponding implementation for situational awareness provided the solution for having the perception of environmental elements and events with respect to time or space. Events detection and the comprehension of their meaning, and the projection of their status after some variable has changed, such as time, or some other variable, such as a predetermined event has been supported by the sensing model and sensing framework solution. This has played an important role also in the realization of industrial scenario (c.f. section 7.2) of project planning in construction sector. The technological solution developed for Consulgal allowed the supervisor in the dimension of being aware of what is happening in the vicinity of the ongoing projects to understand how information, events, and different stakeholders' actions will impact goals and objectives, both immediately and in the future.

9.2. FUTURE WORKS

The technologies being developed allow explorations to be foreseen for the presented research results. Scientific developments and their potential exploitation for industrial cases are envisioned by further enhancements of the technical solutions presented here. From the technical point of view, there are three clear directions:

- For the theoretical model that has been presented in 6.1, it's necessary to make some improvements to provide a more complete model of eCPS in the sense that this research work doesn't take into consideration of formulating consistencies and stabilities of the complex eCPS system that can be formed by aggregating simpler eCPS models developed by using the formulated model. Obviously this is an important future research work, because of the growing trend of distributed computing and hence in future the complex CPSs will of course be composed of simpler CPSs. The theoretical foundation for computing local and global consistencies is an important future work.
- For global sensing framework presented in section 7.1, it is necessary to implement the functionality for providing meta-data for sensors and eventually virtual sensors by using semantic web technologies. The reference ontology to be used for this meta-modelling can be either sensor-ml or ontologies form IoT-A. This is an important functionality to be added and will have direct impact on other features specifically for automatic search and discovery of sensorial objects available in the industrial environment. At the same time the current implementation doesn't have the implementation for reinforcement sensing as described in section 6.1.4. This an interesting and important future works to achieve fault tolerant sensorial networks.
- For the situational awareness framework presented in section 7.2, it's necessary to work further to implement reasoning over real-time data to enable situation detection under incomplete information, by making use of probabilistic modelling. At the same time it's important to note that the presented result has been tested under a very controlled environment where the situations were generated by the activities performed by the users. It's necessary to test the presented solution under the scenarios where the situations are generated during the operation cycles of the machines itself. This is a huge task to undertake considering the permission required to the obtained from industries.
- For the evolutionary algorithm presented in section 7.3, the first task will towards the development of generic domain ontology with EA entities for a wider case of industrial domains such as collaborative supply chain, manufacturing etc. Current research result or the developed ontology only captures a very small business scenario for the construction industry. This is important to test the full operational complexity of the presented solution. At the same time, in order to deal with

situations where the optimization problem requires almost real-time solution it's necessary to work towards the parallel implementation of the algorithm.

- In the overall paradigm of self-evolution, the technical implementation has not been tested well for this characteristic. In future research activities it's expected to take this challenge to implement systems that can show self-evolution characteristics autonomously. In the scope of this work, self-evolution characteristics has been implemented only to the point of detection of evolutionary path and providing necessary suggestions for necessary configuration changes. The implementation is still performed with the intervention of human user by utilizing the dynamic reconfiguration framework.

On the whole, the research work developed and reported within this dissertation leaves some interesting open opportunities for the Future that hopefully will be pursued, by us and by others, and that will improve eCPS model and associated technological solutions, that can make further contributions towards the big challenge of achieving robust CPS.

10. REFERENCES

[1] G. Fischer, Complex systems: Why do they need to evolve and how can evolution be supported, Artif. Intell. Struct. Eng. SE - 8. 1454 (1998) 92–112. doi:10.1007/BFb0030446.

[2] M. Greenbaum, J. and Kyng, Design at work: Cooperative design of computer systems, Educ. Technol. Res. Dev. 42 (1994) 97–99. doi:10.1007/BF02298175.

[3] B. Curtis, K. Herb, I. Neil, A field study of the software design process for large systems, Commun. ACM. 31 (1988) 1268–1287. doi:http://doi.acm.org/10.1145/50087.50089.

[4] P. Guturu, B. Bhargava, Cyber-Physical Systems : A Confluence of Cutting Edge Technological Streams, in: Int. Conf. Adv. Comput. Commun., 2011. https://www.cs.purdue.edu/homes/bb/CPSReviewPaper.pdf.

[5] NIST, Strategic R&D Oportunities for 21st Century Cyber-Physical Systems, 2013. http://www.nist.gov/el/upload/12-Cyber-Physical-Systems020113_final.pdf (accessed October 23, 2014).

[6] Energetics Incorporated, Foundations for Innovation in Cyber-Physical Systems, Work. Report. Natl. Inst. Stand. Technol. (2013). doi:10.1007/s13398-014-0173-7.2.

[7] R.R. Rajkumar, Cyber Physical Systems: A Natural Convergence of Engineering and Computer Science, Triangle Comput. Sci. Disting. Lect. Ser. (2010). http://www.youtube.com/watch?v=7wvq7pVjdJA.

[8] H. Thompson, R. Paulen, M. Reniers, C. Sonntag, S. Engell, Analysis of the State-of-the-Art and Future Challenges in Cyber-physical Systems of Systems, 2015. http://www.cpsos.eu/wp-content/uploads/2015/02/D2-4-State-of-the-art-and-future-challenges-in-cyber-physical-systems-of-2.pdf.

[9] A. Colombo, S. Karnouskos, Towards the factory of the future: A service-oriented cross-layer infrastructure, ICT Shap. World A Sci. View, Eur. Telecommun. Stand. Inst. (2009) 65–81. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.310.4484&rep=rep1&type=pdf (accessed July 23, 2014).

[10] ACATECH – National Academy of Science and Engineering, Cyber-Physical Systems - Driving force for innovation in mobility, health, energy and production, acatech -- National Academy of Science and Engineering, Munich, Germany, 2011. http://www.acatech.de/fileadmin/user_upload/Baumstruktur_nach_Website/Acatech/root/de/Publikationen/Stellungnahmen/acatech_POSITION_CPS_Englisch_WEB.pdf (accessed July 23, 2014).

[11] CyPhERS Cyber-physical European Roadmap & Strategy, CPS : State of the Art, 2014. http://www.cyphers.eu/sites/default/files/D5.1.pdf.

[12] E.K. Wang, Y. Ye, X. Xu, S.M. Yiu, L.C.K. Hui, K.P. Chow, Security Issues and Challenges for Cyber Physical System, 2010 IEEE/ACM Int'l Conf. Green Comput. Commun. Int'l Conf. Cyber, Phys. Soc. Comput. (2010) 733–738. doi:10.1109/GreenCom-CPSCom.2010.36.

[13] M. Anand, E. Cronin, M. Sherr, M. Blaze, Z. Ives, I. Lee, Security Challenges in Next Generation Cyber Physical Systems, Beyond SCADA Networked Embed. Control Cyber Phys. Syst. 41 (2006). http://people.cs.ksu.edu/~danielwang/Investigation/CPS_Security_threat/79e4151082fc028b32.pdf.

[14] B. Bagheri, Big future for cyber-physical manufacturing systems, (2015). http://www.designworldonline.com/big-future-for-cyber-physical-manufacturing-

systems/#_.

[15] J. Rowley, The wisdom hierarchy: representations of the DIKW hierarchy, J. Inf. Sci. 33 (2007) 163–180. doi:10.1177/0165551506070706.

[16] M. Fricke, The knowledge pyramid: a critique of the DIKW hierarchy, J. Inf. Sci. 35 (2008) 131–142. doi:10.1177/0165551508094050.

[17] F. Heylighen, Mediator Evolution: a general scenario for the origin of dynamical hierarchies, in: Worldviews, Sci. Us. (Singapore World Sci., 2006: pp. 1–22. http://pcp.vub.ac.be/Papers/MediatorEvolution.pdf (accessed July 22, 2014).

[18] N. Wiener, Cybernetics or control and communication in the animal and the machine, The MIT Press, 1949. doi:10.1037/h0051026.

[19] T.G. Sugar, V. Kumar, Design and Control of a Compliant Parallel Manipulator, J. Mech. Des. 124 (2002) 676. doi:10.1115/1.1517568.

[20] E. Feigenbaum, P. McCorduck, The Fifth Generation: Artificial Intelligence and Japan's Computer Challenge to the World, Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1983.

[21] T.J. Laffey, P.A. Cox, J.L. Schmidt, S.M. Kao, J.Y. Readk, Real-Time Knowledge-Based Systems, AI Mag. 9 (1988) 27. doi:10.1609/aimag.v9i1.660.

[22] FInES Cluster, Position Paper on Orientations for FP8: A European Innovation Partnership for Catalysing the Competitiveness of European Enterprises, 2011. http://cordis.europa.eu/fp7/ict/enet/documents/fines-position-paper-fp8-orientations-final.pdf.

[23] G. Santucci, C. Martinez, D. Vlad-câlcic, The Sensing Enterprise, in: FInES Work. FIA 2012, 2012: pp. 1–14. http://www.theinternetofthings.eu/sites/default/files/%5Buser-name%5D/Sensing-enterprise.pdf.

[24] B. Krogh, E. Lee, I. Lee, A. Mok, R. Rajkumar, L.R. Sha, A.S. Vincentelli, K. Shin, J. Stankovic, J. Sztipanovits, Cyber-Physical Systems, Executive Summary, 2008. http://iccps.acm.org/2013/_doc/CPS-Executive-Summary.pdf (accessed July 22, 2014).

[25] R. Baheti, H. Gill, Cyber-physical Systems, Impact Control Technol. (2011) 161--166. doi:10.1145/1795194.1795205.

[26] R. Chiong, T. Weise, Z. Michalewicz, Variants of Evolutionary Algorithms for Real-World Applications, Springer Berlin Heidelberg, Berlin, Heidelberg, 2012. doi:10.1007/978-3-642-23424-8.

[27] Z. Michalewicz, Quo Vadis , Evolutionary Computation ? On a Growing Gap between Theory and Practice, (2012) 98–121. https://pdfs.semanticscholar.org/ef96/9f411ece3c424cd4fbae35498c033860fdd3.pdf.

[28] M. Dubreuil, C. Gagne, M. Parizeau, Analysis of a master-slave architecture for distributed evolutionary computations, IEEE Trans. Syst. Man Cybern. Part B. 36 (2006) 229–235. doi:10.1109/TSMCB.2005.856724.

[29] L. Camarinha-Matos, Course: Scientific Research Methodologies and Techniques, (n.d.). https://sites.google.com/a/uninova.pt/cam/teaching/srmt.

[30] M. Balazinska, A. Deshpande, M.J. Franklin, P.B. Gibbons, J. Gray, M. Hansen, M. Liebhold, S. Nath, A. Szalay, V. Tao, Data Management in the Worldwide Sensor Web, IEEE Pervasive Comput. 6 (2007) 30–40. doi:10.1109/MPRV.2007.27.

[31] J. Losos, What Is Evolution?, 2013. http://press.princeton.edu/chapters/i10100.pdf.

[32] L.G. Valiant, Evolvability, J. ACM. 56 (2009) 1–21. doi:10.1145/1462153.1462156.

[33] L.G. Valiant, A theory of the learnable, Commun. ACM. 27 (1984) 1134–1142. doi:10.1145/1968.1972.

[34] C.L. Cummins, S.S. Demastes, M.S. Hafner, Evolution: Biological education's under-researched unifying theme, J. Res. Sci. Teach. 31 (2007) 445–448. doi:10.1002/tea.3660310502.

[35] E. Mayr, This is biology : The Science of the living world, 1999. http://people.umass.edu/sdestef/NRC 601/Mayr 1997b.pdf.

[36] U. Kutschera, K.J. Niklas, The modern theory of biological evolution: an expanded synthesis, Naturwissenschaften. 91 (2004) 255–276. doi:10.1007/s00114-004-0515-y.

[37] E. Mayr, W.B. Provine, The Evolutionary Synthesis: Perspectives on the Unification of Biology, Harvard University Press, 1998.

[38] R.L. Carroll, Towards a new evolutionary synthesis, Trends Ecol. Evol. 15 (2000) 27–32. doi:10.1016/S0169-5347(99)01743-7.

[39] R.L. Carroll, Evolution of the capacity to evolve, J. Evol. Biol. 15 (2002) 911–921. doi:10.1046/j.1420-9101.2002.00455.x.

[40] J.G. Fleagle, The unfused synthesis, Evol. Anthropol. Issues, News, Rev. 10 (2001) 191–191. doi:10.1002/evan.10013.

[41] J.S. Gould, The Structure of Evolutionary Theory, 2003. doi:10.1086/379461.

[42] J. Panno, The cell : evolution of the first organism, 2005. http://www.loc.gov/catdir/toc/ecip0412/2003025841.html.

[43] K.J. Niklas, Plant Biomechanics: An engineering approach to plant form and function, 1992. doi:10.2307/2807461.

[44] K.J. Niklas, Modeling fossil plant form-function relationships: a critique, Paleobiology. 26 (2000) 289–304. doi:10.1666/0094-8373(2000)26[289:MFPFRA]2.0.CO;2.

[45] T. Back, U. Hammel, H.P. Schwefel, Evolutionary computation: Comments on the history and current state, IEEE Trans. Evol. Comput. 1 (1997) 3–17. doi:10.1109/4235.585888.

[46] J.T. Alander, An Indexed Bibliography of Genetic Algorithms in Signal and Image Processing, 2012. http://lipas.uwasa.fi/~TAU/reports/report94-1/gaSIGNALbib.pdf.

[47] C. Darwin, On the origin of species by means of natural selection, or the preservation of favoured races in the struggle for life, D. Appleton and Company, 1859. http://darwin-online.org.uk/converted/pdf/1861_OriginNY_F382.pdf.

[48] A. Abraham, Evolutionary Computation, in: Handb. Meas. Syst. Des., John Wiley & Sons, Ltd, Chichester, UK, 2005: p. 272. doi:10.1002/0471497398.mm423.

[49] J.R. Sampson, Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence, MIT Press, 1992.

[50] H.-G. Beyer, H.-P. Schwefel, Evolution strategies -- A comprehensive introduction, Nat. Comput. 1 (2002) 3–52. doi:10.1023/A:1015059928466.

[51] N. Hansen, D. V Arnold, A. Auger, Evolution Strategies, in: J. Kacprzyk, W. Pedrycz (Eds.), Springer Handb. Comput. Intell., Springer Berlin Heidelberg, Berlin, Heidelberg, 2015: pp. 871–898. doi:10.1007/978-3-662-43505-2_44.

[52] L.J. Fogel, A.J. Owens, M.J. Walsh, Artificial Intelligence through Simulated Evolution, Wiley. (1997) 27–38. doi:citeulike-article-id:431795.

[53] D.M. Rocke, Z. Michalewicz, Genetic Algorithms + Data Structures = Evolution Programs, J. Am. Stat. Assoc. 95 (2000) 347. doi:10.2307/2669583.

[54] R. Chiong, T. Weise, Z. Michalewicz, Variants of Evolutionary Algorithms for Real-

World Applications (Google eBook), Springer Publishing Company, Incorporated, 2011.

[55] M. Mitchell, An introduction to genetic algorithms, Comput. Math. with Appl. 32 (1996) 133. doi:10.1016/S0898-1221(96)90227-8.

[56] P.G. Aggarwal, Shaifali, Richa Garg, A Review Paper on Different Encoding Schemes used in Genetic Algorithms, Adv. Res. Comput. Sci. Softw. Eng. 4 (2014) 596–600. http://www.ijarcsse.com/docs/papers/Volume_4/1_January2014/V4I1-0373.pdf.

[57] H. Hofmeyer, J.M. Davila Delgado, Coevolutionary and genetic algorithm based building spatial and structural design, Artif. Intell. Eng. Des. Anal. Manuf. 29 (2015) 351–370. doi:10.1017/S0890060415000384.

[58] K.A. De Jong, W.M. Spears, A formal analysis of the role of multi-point crossover in genetic algorithms, Ann. Math. Artif. Intell. 5 (1992) 1–26. doi:10.1007/BF01530777.

[59] L.J. Fogel, A.J. Owens, M.J. Walsh, Artificial Intelligence through Simulated Evolution, Wiley. (1997) 27–38. doi:citeulike-article-id:431795.

[60] D.B. Fogel, L.J. Fogel, J.W. Atmar, Meta-evolutionary programming, in: Conf. Rec. Twenty-Fifth Asilomar Conf. Signals, Syst. Comput., IEEE Comput. Soc. Press, 1991: pp. 540–545. doi:10.1109/ACSSC.1991.186507.

[61] H.-P. Schwefel, Evolution and Optimum Seeking, Sixth-Generation Comput. Technol. Ser. (1995) ix, 444. http://www.worldcat.org/oclc/30701094.

[62] J.R. Koza, Genetic Programming: A Paradigm for Genetically Breeding Populations of Computer Programs to Solve Problems, Stanford University, Stanford, CA, USA, 1990.

[63] R. Poli, W.B. Langdon, N.F. McPhee, A Field Guide to Genetic Programing, 2008. http://www.essex.ac.uk/wyvern/2008-04/Wyvern April 08 7126.pdf.

[64] T. Jansen, Analyzing Evolutionary Algorithms, Springer Berlin Heidelberg, Berlin, Heidelberg, 2013. doi:10.1007/978-3-642-17339-4.

[65] A.L. Parrill, Introduction to Evolutionary Algorithms, in: Evol. Algorithms Mol. Des., 2008: pp. 1–13. doi:10.1002/9783527613168.ch1.

[66] K.A. De Jong, Evolutionary computation: A unified approach, The MIT Press, 2006.

[67] R. Kruse, C. Borgelt, F. Klawonn, C. Moewes, M. Steinbrecher, P. Held, Computational Intelligence, Springer London, London, 2013. doi:10.1007/978-1-4471-5013-8.

[68] A.E. Eiben, J. Smith, From evolutionary computation to the evolution of things, Nature. 521 (2015) 476–482. doi:10.1038/nature14544.

[69] T. Ikemoto, M. Kawasaki, T. Kato, R. Takemasa, T. Tani, K. Taniuchi, T. Ushida, Dangerous cervical radiculopathy by Lemierre's syndrome, J. Orthop. Sci. 17 (2012) 663–666. doi:10.1007/s00776-011-0105-8.

[70] N. Axten, A. Newell, H. a. Simon, Human Problem Solving., Contemp. Sociol. 2 (1973) 169. doi:10.2307/2063712.

[71] T.P. Runarsson, Evolutionary Problem Solving, 2000. https://notendur.hi.is/tpr/papers/eps.pdf.

[72] W.B. Powell, Approximate Dynamic Programming, in: Dyn. Program. Optim. Control, 2011: pp. 329–552. http://web.mit.edu/dimitrib/www/dpchapter.pdf.

[73] G. Oddi, A. Pietrabissa, A distributed multi-path algorithm for wireless ad-hoc networks based on Wardrop routing, in: 21st Mediterr. Conf. Control Autom., IEEE, 2013: pp. 930–935. doi:10.1109/MED.2013.6608833.

[74] S.M. Scheiner, The genetics of phenotypic plasticity. VII. Evolution in a spatially-structured environment, J. Evol. Biol. 11 (1998) 303. doi:10.1007/s000360050090.

[75] D.B. Fogel, Evolutionary computation: toward a new philosophy of machine intelligence, 2006. http://samples.sainsburysebooks.co.uk/9780471749202_sample_383526.pdf.

[76] Y. Xu, R. Qu, Solving multi-objective multicast routing problems by evolutionary multi-objective simulated annealing algorithms with variable neighbourhoods, J. Oper. Res. Soc. 62 (2011) 313–325. doi:10.1057/jors.2010.138.

[77] Y. Jin, Evolutionary Computation in Dynamic and Uncertain Environments, Springer Berlin Heidelberg, Berlin, Heidelberg, 2007. doi:10.1007/978-3-540-49774-5.

[78] I. (Ed.). Vlahavas, Artificial Intelligence for Advanced Problem Solving Techniques, IGI Global, 2008. doi:10.4018/978-1-59904-705-8.

[79] G.S. Hornby, J.D. Lohn, D.S. Linden, Computer-Automated Evolution of an X-Band Antenna for NASA's Space Technology 5 Mission, Evol. Comput. 19 (2011) 1–23. doi:10.1162/EVCO_a_00005.

[80] A. Arias-Montano, C.A.C. Coello, E. Mezura-Montes, Multiobjective Evolutionary Algorithms in Aeronautical and Aerospace Engineering, IEEE Trans. Evol. Comput. 16 (2012) 662–694. doi:10.1109/TEVC.2011.2169968.

[81] J. Besnard, G.F. Ruda, V. Setola, K. Abecassis, R.M. Rodriguiz, X.-P. Huang, S. Norval, M.F. Sassano, A.I. Shin, L. a Webster, F.R.C. Simeons, L. Stojanovski, A. Prat, N.G. Seidah, D.B. Constam, G.R. Bickerton, K.D. Read, W.C. Wetsel, I.H. Gilbert, B.L. Roth, A.L. Hopkins, Automated design of ligands to polypharmacological profiles, Nature. 492 (2012) 215–220. doi:10.1038/nature11691.

[82] M. Kapur, J. Voiklis, C.K. Kinzer, Problem solving as a complex, evolutionary activity: A methodological framework for analyzing problem-solving processes in a computer supported collaborative environment, in: Proc. Th 2005 Conf. Comput. Support Collab. Learn. Learn. 2005 Next 10 Years!, 2005: pp. 252–261. http://dl.acm.org/citation.cfm?id=1149293.1149326.

[83] D.H. Wolpert, W.G. Macready, No free lunch theorems for optimization, IEEE Trans. Evol. Comput. 1 (1997) 67–82. doi:10.1109/4235.585893.

[84] A. Ray, Autonomous perception and decision-making in cyber-physical systems, in: 2013 8th Int. Conf. Comput. Sci. Educ., IEEE, 2013: pp. 1–10. doi:10.1109/ICCSE.2013.6554173.

[85] M.M. Lehman, J.F. Ramil, Software evolution—Background, theory, practice, Inf. Process. Lett. 88 (2003) 33–44. doi:10.1016/S0020-0190(03)00382-X.

[86] H. Barringer, D. Rydeheard, D. Gabbay, A Logical Framework for Monitoring and Evolving Software Components, in: First Jt. IEEE/IFIP Symp. Theor. Asp. Softw. Eng. (TASE '07), IEEE, 2007: pp. 273–282. doi:10.1109/TASE.2007.4.

[87] S.-W. Cheng, D. Garlan, B. Schmerl, Making Self-Adaptation an Engineering Reality, in: Ö. Babaoglu, M. Jelasity, A. Montresor, C. Fetzer, S. Leonardi, A.P.A. van Moorsel, M. van Steen (Eds.), Self-Star Prop. Complex, Springer, 2005: pp. 158–173. doi:10.1007/11428589_11.

[88] M.M. Lehman, J.F. Ramil, G. Kahen, Evolution as a Noun and Evolution as a Verb, Proc. Work. Softw. Organ. Co-Evolution. (2000) 14–15. doi:10.1.1.35.759.

[89] F. Heylighen, The science of self-organization and adaptativity, in: Encycl. Life Support Syst. EOLSS, 2001: pp. 253--280.

[90] R. Der, On the Role of Embodiment for Self-Organizing Robots: Behavior As Broken Symmetry, in: Guid. Self-Organization Inception, 2014: pp. 193–221. doi:10.1007/978-3-642-53734-9_7.

[91] W.R. Ashby, Design for a brain : the origin of adaptive behaviour, Chapman and Hall,

London, 1960. http://opac.inria.fr/record=b1096620.

[92] O. Hoftberger, R. Obermaisser, Ontology-based runtime reconfiguration of distributed embedded real-time systems, in: 16th IEEE Int. Symp. Object/component/service-Oriented Real-Time Distrib. Comput. (ISORC 2013), IEEE, 2013: pp. 1–9. doi:10.1109/ISORC.2013.6913205.

[93] E. Bartocci, R. Grosu, A. Karmarkar, Runtime Verification, Springer Berlin Heidelberg, Berlin, Heidelberg, 2013. doi:10.1007/978-3-642-35632-2.

[94] K. Nie, T. Yue, S. Ali, Towards a search-based interactive configuration of cyber physical system product lines, in: CEUR Workshop Proc., 2013: pp. 71–75. http://ceur-ws.org/Vol-1115/poster4.pdf (accessed October 30, 2014).

[95] S.R. Bowman, C. Potts, C.D. Manning, Recursive Neural Networks Can Learn Logical Semantics, in: Proc. 3rd Work. Contin. Vector Sp. Model. Their Compos., Association for Computational Linguistics, Stroudsburg, PA, USA, 2015: pp. 12–21. doi:10.18653/v1/W15-4002.

[96] M.L. Caliusco, G. Stegmayer, Emergent Web Intelligence: Advanced Semantic Technologies, Springer London, London, 2010. doi:10.1007/978-1-84996-077-9.

[97] T. Mikolov, G. Zweig, Context dependent recurrent neural network language model, in: 2012 IEEE Spok. Lang. Technol. Work., IEEE, 2012: pp. 234–239. doi:10.1109/SLT.2012.6424228.

[98] R. Mikut, J. Jäkel, L. Gröll, Interpretability issues in data-based learning of fuzzy systems, Fuzzy Sets Syst. 150 (2005) 179–197. doi:10.1016/j.fss.2004.06.006.

[99] E. Lughofer, E. Hüllermeier, E. Klement, Improving the interpretability of data-driven evolving fuzzy-systems, in: Proc. - 4th Conf. Eur. Soc. Fuzzy Log. Technol. 11th French Days Fuzzy Log. Appl. EUSFLAT-LFA 2005 Jt. Conf., 2005: pp. 28–33. http://www.eusflat.org/proceedings/EUSFLAT-LFA_2005/papers definitivos/S104-03.pdf (accessed October 30, 2014).

[100] P. Angelov, D.P. Filev, N. Kasabov, Evolving intelligent systems: methodology and applications, John Wiley & Sons, 2010. https://books.google.com.br/books?hl=pt-BR&lr=&id=mhYTIq7vouoC&oi=fnd&pg=PR5&dq=Evolving+Intelligent+Systems:+Methodology+and+Applications&ots=S2nYU73d5L&sig=AJGn_s3z4dn4Si9_Vp03OnPLfRU#v=onepage&q&f=false.

[101] M. Raciti, Anomaly Detection and its Adaptation: Studies on Cyber-Physical Systems, 2013. http://www.diva-portal.org/smash/get/diva2:608664/FULLTEXT01.pdf (accessed October 30, 2014).

[102] U. Tangen, From evolving software towards models of dynamically self-assembling processing systems, Proc. ECCS. (2006) 1–15. http://www.cabdyn.ox.ac.uk/complexity_PDFs/ECCS06/Conference_Proceedings/PDF/p85.pdf (accessed April 22, 2013).

[103] B.H.C. Cheng, J.M. Atlee, Research Directions in Requirements Engineering, in: Futur. Softw. Eng. (FOSE '07), IEEE, Washington, DC, USA, 2007: pp. 285–303. doi:10.1109/FOSE.2007.17.

[104] M. Sabetzadeh, S. Easterbrook, View merging in the presence of incompleteness and inconsistency, Requir. Eng. 11 (2006) 174–193. doi:10.1007/s00766-006-0032-y.

[105] D. Garlan, S. Cheng, B. Schmerl, Increasing System Dependability through Architecture-Based Self-Repair, in: Archit. Dependable Syst., 2003: pp. 61–89. doi:10.1007/3-540-45177-3_3.

[106] G.A. Dumont, M. Huzmezan, Concepts, methods and techniques in adaptive control, in:

Proc. 2002 Am. Control Conf. (IEEE Cat. No.CH37301), IEEE, 2002: pp. 1137–1150 vol.2. doi:10.1109/ACC.2002.1023173.

[107] F.J. Rammig, S. Grosbrink, K. Stahl, Y. Zhao, Designing Self-Adaptive Embedded Real-Time Software -- Towards System Engineering of Self-Adaptation, in: 2014 Brazilian Symp. Comput. Syst. Eng., IEEE, 2014: pp. 37–42. doi:10.1109/SBESC.2014.15.

[108] D. Cerri, S. Terzi, Proposal of a toolset for the improvement of industrial systems' lifecycle sustainability through the utilization of ICT technologies, Comput. Ind. 81 (2016) 47–54. doi:10.1016/j.compind.2015.09.003.

[109] W.N. Robinson, A requirements monitoring framework for enterprise systems, Requir. Eng. 11 (2006) 17–41. doi:10.1007/s00766-005-0016-3.

[110] N. Bencomo, J. Whittle, P. Sawyer, A. Finkelstein, E. Letier, Requirements reflection, in: Proc. 32nd ACM/IEEE Int. Conf. Softw. Eng. - ICSE '10, ACM Press, New York, New York, USA, 2010: p. 199. doi:10.1145/1810295.1810329.

[111] J. Andersson, R. de Lemos, S. Malek, D. Weyns, Modeling Dimensions of Self-Adaptive Software Systems, in: B.H.C. Cheng, R. de Lemos, H. Giese, P. Inverardi, J. Magee (Eds.), Softw. Eng. Self-Adaptive Syst., Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, Germany, Dagstuhl, Germany, 2009: pp. 27–47. doi:10.1007/978-3-642-02161-9_2.

[112] G. Engels, M. Assmann, Service-Oriented Enterprise Architectures: Evolution of Concepts and Methods, in: 2008 12th Int. IEEE Enterp. Distrib. Object Comput. Conf., IEEE, 2008: pp. xxxiv–xliii. doi:10.1109/EDOC.2008.48.

[113] F. Kon, F. Costa, G. Blair, R.H. Campbell, The case for reflective middleware, Commun. ACM. 45 (2002) 33–38. doi:10.1145/508448.508470.

[114] G. Coulson, G. Blair, P. Grace, F. Taiani, A. Joolia, K. Lee, J. Ueyama, T. Sivaharan, A generic component model for building systems software, ACM Trans. Comput. Syst. 26 (2008) 1–42. doi:10.1145/1328671.1328672.

[115] W.N. Robinson, A requirements monitoring framework for enterprise systems, Requir. Eng. 11 (2006) 17–41. doi:10.1007/s00766-005-0016-3.

[116] T. Bures, I. Gerostathopoulos, P. Hnetynka, J. Keznikl, M. Kit, F. Plasil, DEECo – an Ensemble-Based Component System DEECo – an Ensemble-Based Component System, (2013). http://d3s.mff.cuni.cz/publications/download/D3S-TR-2013-02.pdf.

[117] J. Keznikl, T. Bures, F. Plasil, I. Gerostathopoulos, P. Hnetynka, N. Hoch, Design of ensemble-based component systems by invariant refinement, in: Proc. 16th Int. ACM Sigsoft Symp. Component-Based Softw. Eng. - CBSE '13, ACM Press, New York, New York, USA, 2013: p. 91. doi:10.1145/2465449.2465457.

[118] M. Onori, D.T. Semere, B. Lindberg, Evolvable Systems: An Approach to Self-X Production, in: G.Q. Huang, K.L. Mak, P.G. Maropoulos (Eds.), Proc. 6th CIRP-Sponsored Int. Conf. Digit. Enterp. Technol., Springer Berlin Heidelberg, Berlin, Heidelberg, 2010: pp. 789–802. doi:10.1007/978-3-642-10430-5_61.

[119] D. Semere, J. Barata, M. Onori, Evolvable Assembly Systems: Developments and Advances, in: 2007 IEEE Int. Symp. Assem. Manuf., IEEE, 2007: pp. 282–287. doi:10.1109/ISAM.2007.4288486.

[120] M. Onori, J. Barata Oliveira, Outlook report on the future of European assembly automation, Assem. Autom. 30 (2010) 7–31. doi:10.1108/01445151011016028.

[121] M. Onori, J. Barata, Evolvable Production Systems: New domains within mechatronic production equipment, in: L.M. Camarinha-Matos, P. Pereira, L. Ribeiro (Eds.), 2010 IEEE Int. Symp. Ind. Electron., IEEE, 2010: pp. 2653–2657. doi:10.1109/ISIE.2010.5637827.

[122] S. Ghimire, F. Luis-Ferriera, R. Jardim-Goncalves, T. Nodehi, U. Beijing Jiaotong, E. Chinese Academy of, S. Chinese Mechanical Engineering, E. International Society for Productivity, C. National Natural Science Foundation of, et al., Towards self-evolutionary cyber physical systems, 21st ISPE Inc. Int. Conf. Concurr. Eng. CE 2014. (2014) 547–554. doi:10.3233/978-1-61499-440-4-547.

[123] A. Rajhans, S.-W. Cheng, B. Schmerl, D. Garlan, B.H. Krogh, C. Agbi, A. Bhave, An Architectural Approach to the Design and Analysis of Cyber-Physical Systems, Electron. Commun. EASST. 21 (2009). doi:10.14279/tuj.eceasst.21.286.278.

[124] L. Sha, S. Gopalakrishnan, X. Liu, Q. Wang, Cyber-Physical Systems: A New Frontier, in: Mach. Learn. Cyber Trust, Springer US, Boston, MA, 2009: pp. 3–13. doi:10.1007/978-0-387-88735-7_1.

[125] E. Commission, Cyber-Physical Systems : Uplifting Europe's Innovation Capacity, in: Rep. from Work. Cyber-Physical Syst., 2013. http://ec.europa.eu/information_society/newsroom/cf/dae/document.cfm?doc_id=3948.

[126] Kyoung-Dae Kim, P.R. Kumar, Cyber–Physical Systems: A Perspective at the Centennial, Proc. IEEE. 100 (2012) 1287–1308. doi:10.1109/JPROC.2012.2189792.

[127] T. Sanislav, L. Miclea, Cyber-Physical Systems - Concept, Challenges and Research Areas, J. Control Eng. Appl. Informatics. 14 (2012) 28–33. http://ceai.srait.ro/index.php/ceai/article/view/1292 (accessed July 22, 2014).

[128] A. Ahmad, A. Paul, M.M. Rathore, H. Chang, Smart cyber society: Integration of capillary devices with high usability based on Cyber–Physical System, Futur. Gener. Comput. Syst. 56 (2016) 493–503. doi:10.1016/j.future.2015.08.004.

[129] S. Karnouskos, Cyber-Physical Systems in the SmartGrid, in: 2011 9th IEEE Int. Conf. Ind. Informatics, IEEE, 2011: pp. 20–23. doi:10.1109/INDIN.2011.6034829.

[130] D. Work, A. Bayen, Q. Jacobson, Automotive Cyber Physical Systems in the Context of Human Mobility, in: Natl. Work. High-Confidence Automot. Cyber-Physical Syst., 2008: pp. 3–5. http://bayen.eecs.berkeley.edu/sites/default/files/conferences/cps1.pdf.

[131] H.-A. Kao, W. Jin, D. Siegel, J. Lee, A Cyber Physical Interface for Automation Systems—Methodology and Examples, Machines. 3 (2015) 93–106. doi:10.3390/machines3020093.

[132] O. Sokolsky, Medical Cyber-Physical Systems, in: 2011 18th IEEE Int. Conf. Work. Eng. Comput. Syst., IEEE, 2011: pp. 2–2. doi:10.1109/ECBS.2011.40.

[133] R. Poovendran, Cyber–Physical Systems: Close Encounters Between Two Parallel Worlds [Point of View], Proc. IEEE. 98 (2010) 1363–1366. doi:10.1109/JPROC.2010.2050377.

[134] C. Alippi, Intelligence for Embedded Systems, Springer International Publishing, Cham, 2014. doi:10.1007/978-3-319-05278-6.

[135] B.E. Beckmann, L.M. Grabowski, P.K. McKinley, C. Ofria, Applying digital evolution to the design of self-adaptive software, in: 2009 IEEE Symp. Artif. Life, IEEE, 2009: pp. 100–107. doi:10.1109/ALIFE.2009.4937700.

[136] L. Ribeiro, J. Barata, G. Cândido, M. Onori, Evolvable Production Systems: An Integrated View on Recent Developments, in: G.Q. Huang, K.L. Mak, P.G. Maropoulos (Eds.), Proc. 6th CIRP-Sponsored Int. Conf. Digit. Enterp. Technol., Springer Berlin Heidelberg, Berlin, Heidelberg, 2010: pp. 841–854. doi:10.1007/978-3-642-10430-5_65.

[137] P. Derler, E.A. Lee, A.L. Sangiovanni-vincentelli, Addressing Modeling Challenges in Cyber-Physical Systems, 2011. http://digitalassets.lib.berkeley.edu/techreports/ucb/text/EECS-2011-17.pdf.

[138] W.F. Van Der Vegte, R.W. Vroom, Considering cognitive aspects in designing cyber – physical systems: an emerging need for transdisciplinarity, in: 2013: pp. 41–52. http://tinyurl.com/j4ud5wk.

[139] J.N.O. Holland, Emergence: From Chaos to Order, Perseus Publishing, 1998.

[140] F. Heylighen, The science of self-organization and adaptivity, Encycl. Life Support Syst. 5 (2001) 253–280.

[141] D. Levy, Applications and limitations of complexity theory in organization theory and strategy, Public Adm. Public Policy. (2000) 67–87. http://www.faculty.umb.edu/david_levy/complex00.pdf (accessed October 21, 2014).

[142] M. Mitchell, M. Newman, Complex systems theory and evolution, in: Encycl. Evol., 2002: pp. 1–5. http://web.cecs.pdx.edu/~mm/EncycOfEvolution.pdf (accessed June 26, 2014).

[143] P. Trueba, A. Prieto, F. Bellas, P. Caamaño, R.J. Duro, Self-organization and Specialization in Multiagent Systems through Open-Ended Natural Evolution, in: Lect. Notes Comput. Sci. (Including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics), Springer-Verlag, Berlin, Heidelberg, 2012: pp. 93–102. doi:10.1007/978-3-642-29178-4_10.

[144] A.F. Smullyan, Morris Charles. Signs, language, and behavior. Prentice-Hall, Inc., New York 1946, v + 356 pp., J. Symb. Log. 12 (1947) 49–51. doi:10.2307/2267248.

[145] P. Mittal, Knowledge Extraction based on Evolutionary Learning (KEEL): Analysis of Development Method , Genetic Fuzzy System, Int. J. Comput. Appl. Inf. Technol. 1 (2012) 22–25. http://www.ijcait.com/IJCAIT/117.pdf.

[146] Y. Zhang, P. Louvieris, M. Petrou, Case-Based Reasoning Adaptation for High Dimensional Solution Space, Trans. Case-Based Reason. Multimed. Data. 1 (2008) 21–36. http://www.ibai-publishing.org/journal/issue_cbr/2008_october/cbrmd_1_1_21-36.pdf (accessed October 30, 2014).

[147] B. Fuchs, J. Lieber, A. Mille, A. Napoli, An algorithm for adaptation in case-based reasoning, in: ECAI 2000, 2000: pp. 45–49. http://frontiersinai.com/ecai/ecai2000/pdf/p0045.pdf (accessed October 30, 2014).

[148] P. Seymour, A. Schrijver, R. Diestel, Graph Theory, 2005. doi:10.4171/OWR/2005/03.

[149] M. Clemens, Complexity Illustrations, Ideagram. (2004). http://www.idiagram.com/examples/complexity.html.

[150] D. Kempe, Structure and dynamics of information in networks, 2011. http://www-bcf.usc.edu/~dkempe/teaching/structure-dynamics.pdf.

[151] F. Hu, Y. Lu, A. V. Vasilakos, Q. Hao, R. Ma, Y. Patil, T. Zhang, J. Lu, X. Li, N.N. Xiong, Robust Cyber–Physical Systems: Concept, models, and implementation, Futur. Gener. Comput. Syst. 56 (2016) 449–475. doi:10.1016/j.future.2015.06.006.

[152] E. a. Lee, S. Neuendorffer, M.J. Wirthlin, Actor-Oriented Design of Embedded Hardware and Software Systems, J. Circuits, Syst. Comput. 12 (2003) 231–260. doi:10.1142/S0218126603000751.

[153] A. Benveniste, T. Bourke, B. Caillaud, M. Pouzet, Hybrid Systems Modeling Challenges caused by Cyber-Physical Systems, (2012) 1–22. http://people.rennes.inria.fr/Albert.Benveniste/pub/NIST2012.pdf.

[154] P. Martin, M.B. Egerstedt, Hybrid systems tools for compiling controllers for cyber-physical systems, Discret. Event Dyn. Syst. 22 (2012) 101–119. doi:10.1007/s10626-011-0117-8.

[155] P. Derler, E.A. Lee, A.S. Vincentelli, Modeling Cyber–Physical Systems, Proc.

IEEE. 100 (2012) 13–28. doi:10.1109/JPROC.2011.2160929.

[156] J.C. Jensen, D.H. Chang, E.A. Lee, A model-based design methodology for cyber-physical systems, in: 2011 7th Int. Wirel. Commun. Mob. Comput. Conf., IEEE, 2011: pp. 1666–1671. doi:10.1109/IWCMC.2011.5982785.

[157] M. Hailperin, B. Kaiser, K. Knight, Concrete Abstractions: An Introduction to Computer Science Using Scheme, 1999. http://www.gustavus.edu/+max/concrete-abstractions.html.

[158] L. Libkin, Elements of Finite Model Theory, Springer Berlin Heidelberg, Berlin, Heidelberg, 2004. doi:10.1007/978-3-662-07003-1.

[159] A. Diller, R. Docherty, Z and Abstract Machine Notation: a comparison, in: Proc. 8th Z User Meet. - ZUM'94, 1994: p. p.250-63. http://www.cantab.net/users/antoni.diller/papers/s2.pdf.

[160] E.A. Lee, H. Zheng, Operational Semantics of Hybrid Systems, in: Hybrid Syst. Comput. Control 8th Int. Work. HSCC 2005, Zurich, Switzerland, March 9-11, 2005. Proc., 2005: pp. 25–53. doi:10.1007/978-3-540-31954-2_2.

[161] E. a Lee, Finite State Machines and Modal Models in Ptolemy II, 2009. http://digitalassets.lib.berkeley.edu/techreports/ucb/text/EECS-2009-151.pdf.

[162] E. a Lee, No Title, Ann. Softw. Eng. 7 (1999) 25–45. doi:10.1023/A:1018998524196.

[163] E.A. Lee, T.M. Parks, Dataflow process networks, Proc. IEEE. 83 (1995) 773–801. doi:10.1109/5.381846.

[164] J. Stephen Topper, N.C. Horner, Model-based systems engineering in support of complex systems development, Johns Hopkins APL Tech. Dig. (Applied Phys. Lab. 32 (2013) 419–432. http://www.jhuapl.edu/techdigest/TD/td3201/32_01-Topper.pdf.

[165] G. Hackmann, Weijun Guo, Guirong Yan, Zhuoxiong Sun, Chenyang Lu, S. Dyke, Cyber-Physical Codesign of Distributed Structural Health Monitoring with Wireless Sensor Networks, IEEE Trans. Parallel Distrib. Syst. 25 (2014) 63–72. doi:10.1109/TPDS.2013.30.

[166] V.K. Singh, R. Jain, Situation based control for cyber-physical environments, in: MILCOM 2009 - 2009 IEEE Mil. Commun. Conf., IEEE, 2009: pp. 1–7. doi:10.1109/MILCOM.2009.5380000.

[167] Y. Zhu, R. Cartwright, A. Ames, R. Bhattacharya, E. Westbrook, J. Inoue, A. Chapoutot, C. Salama, M. Peralta, T. Martin, W. Taha, M. O'Malley, Mathematical equations as executable models of mechanical systems, in: Proc. 1st ACM/IEEE Int. Conf. Cyber-Physical Syst. - ICCPS '10, ACM Press, New York, New York, USA, 2010: p. 1. doi:10.1145/1795194.1795196.

[168] B. Selic, The pragmatics of model-driven development, IEEE Softw. 20 (2003) 19–25. doi:10.1109/MS.2003.1231146.

[169] C. Gershenson, Design and Control of Self-organizing Systems, 2007. http://cogprints.org/5442/1/thesis.pdf (accessed July 21, 2014).

[170] M. Shaw, D. Garlan, Book review: Software Architecture: Perspectives on an Emerging Discipline, Manag. Syst. Dev. 17 (1997) 6.

[171] D.E. Perry, A.L. Wolf, Foundations for the study of software architecture, ACM SIGSOFT Softw. Eng. Notes. 17 (1992) 40–52. doi:10.1145/141874.141884.

[172] L. Bass, P. Clements, R. Kazman, Software Architecture in Practice, Addison-Wesley Longman Publishing Co., Inc., 2012. http://tinyurl.com/zzr5w89 (accessed October 24, 2014).

[173] R. James, J. Ivar, B. Grady, The unified modeling language reference manual, Read.

Addison Wesley. (1999). http://msdl.cs.mcgill.ca/people/tfeng/docs/The Unified Modeling Language Reference Manual.pdf (accessed October 24, 2014).

[174] OMG, OMG Systems Modeling Language (OMG SysML), 2015. http://www.omg.org/spec/SysML/1.2/PDF/.

[175] P.H. Feiler, D.P. Gluch, J.J. Hudak, The Architecture Analysis & Design Language (AADL): An Introduction, 2006. https://people.cs.clemson.edu/~johnmc/courses/cpsc875/resources/06tn011.pdf (accessed October 24, 2014).

[176] B. Schmerl, D. Garlan, AcmeStudio: supporting style-centered architecture development, in: Proceedings. 26th Int. Conf. Softw. Eng., IEEE Comput. Soc, 2004: pp. 704–705. doi:10.1109/ICSE.2004.1317497.

[177] J. Hudak, P. Feiler, Developing AADL Models for Control Systems: A Practitioner's Guide, 2007. http://www.sei.cmu.edu/reports/07tr014.pdf.

[178] K. Bauer, A New Modelling Language for Cyber-physical Systems, 2012. https://es.cs.uni-kl.de/publications/datarsg/Baue12.pdf.

[179] N. Saeedloei, Modeling and Verificaion of Real-time and Cyber-Physical Systems, 2011. http://www.utdallas.edu/~gupta/nedathesis.pdf.

[180] A. Metzger, K. Pohl, Software product line engineering and variability management: achievements and challenges, in: Proc. Futur. Softw. Eng. - FOSE 2014, ACM Press, New York, New York, USA, 2014: pp. 70–84. doi:10.1145/2593882.2593888.

[181] G. Cardone, L. Foschini, P. Bellavista, A. Corradi, C. Borcea, M. Talasila, R. Curtmola, Fostering participaction in smart cities: a geo-social crowdsensing platform, IEEE Commun. Mag. 51 (2013) 112–119. doi:10.1109/MCOM.2013.6525603.

[182] Networked European Software and Services Initiative, Cyber Physical Systems Software , Services , Cloud and Data, 2015. http://www.nessi-europe.eu/Files/Private/NESSI_CPS_White_Paper_issue_1.pdf.

[183] S. Wiesner, C. Gorldt, M. Soeken, K.-D. Thoben, R. Drechsler, Requirements Engineering for Cyber-Physical Systems, in: Adv. Prod. Manag. Syst. Innov. Knowledge-Based Prod. Manag. a Glob. World, 2014: pp. 281–288. doi:10.1007/978-3-662-44739-0_35.

[184] C. Berger, Accelerating Regression Testing for Scaled Self-Driving Cars with Lightweight Virtualization -- A Case Study, in: 2015 IEEE/ACM 1st Int. Work. Softw. Eng. Smart Cyber-Physical Syst., IEEE, 2015: pp. 2–7. doi:10.1109/SEsCPS.2015.9.

[185] X. Zheng, C. Julien, Verification and Validation in Cyber Physical Systems: Research Challenges and a Way Forward, in: 2015 IEEE/ACM 1st Int. Work. Softw. Eng. Smart Cyber-Physical Syst., IEEE, 2015: pp. 15–18. doi:10.1109/SEsCPS.2015.11.

[186] S. Mitsch, A. Platzer, ModelPlex: Verified Runtime Validation of Verified Cyber-Physical System Models, in: Lect. Notes Comput. Sci. (Including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics), 2014: pp. 199–214. doi:10.1007/978-3-319-11164-3_17.

[187] P. Wang, Y. Xiang, S.H. Zhang, Cyber-Physical System Components Composition Analysis and Formal Verification Based on Service-Oriented Architecture, in: 2012 IEEE Ninth Int. Conf. E-Bus. Eng., IEEE, 2012: pp. 327–332. doi:10.1109/ICEBE.2012.60.

[188] V. Szalvay, An introduction to agile software development, 2004. http://www.danube.com/docs/Intro_to_Agile.pdf.

[189] J. Blankenship, M. Bussa, S. Millett, The Art of Agile Development, in: Pro Agil. .NET Dev. with Scrum, O'Reilly, 2011: pp. 1–11. doi:10.1007/978-1-4302-3534-7.

[190] E. Stelzmann, Contextualizing agile systems engineering, IEEE Aerosp. Electron. Syst.

Mag. 27 (2012) 17–22. doi:10.1109/MAES.2012.6226690.

[191] Technology Institute, The nine strategies that lead to successful embedded software development, 2013. https://www.pwc.com/us/en/technology/publications/assets/pwc-agile-embedded-software-development.pdf.

[192] J.A. Sharp, Data oriented program design, ACM SIGPLAN Not. 15 (1980) 44–57. doi:10.1145/947706.947713.

[193] A.H. Eden, A Theory of Object-Oriented Design, Inf. Syst. Front. 4 (2002) 379–391. doi:10.1023/A:1020835709566.

[194] S.R. Madden, M.J. Franklin, J.M. Hellerstein, W. Hong, TinyDB: an acquisitional query processing system for sensor networks, ACM Trans. Database Syst. 30 (2005) 122–173. doi:10.1145/1061318.1061322.

[195] J. King, R. Bose, H. Yang, S. Pickles, A. Helal, Atlas: A Service-Oriented Sensor Platform: Hardware and Middleware to Enable Programmable Pervasive Spaces, in: Proceedings. 2006 31st IEEE Conf. Local Comput. Networks, IEEE, 2006: pp. 630–638. doi:10.1109/LCN.2006.322026.

[196] X. Chu, T. Kobialka, B. Durnota, R. Buyya, Open Sensor Web Architecture: Core Services, in: 2006 Fourth Int. Conf. Intell. Sens. Inf. Process., IEEE, 2006: pp. 98–103. doi:10.1109/ICISIP.2006.4286069.

[197] Hock Beng Lim, Yong Meng Teo, P. Mukherjee, Vinh The Lam, Weng Fai Wong, S. See, Sensor grid: integration of wireless sensor networks and the grid, in: IEEE Conf. Local Comput. Networks 30th Anniv. (LCN'05)l, IEEE, 2005: pp. 91–99. doi:10.1109/LCN.2005.123.

[198] M. Kushwaha, I. Amundson, X. Koutsoukos, S. Neema, J. Sztipanovits, OASiS: A Programming Framework for Service-Oriented Sensor Networks, in: 2007 2nd Int. Conf. Commun. Syst. Softw. Middlew., IEEE, 2007: pp. 1–8. doi:10.1109/COMSWA.2007.382431.

[199] R.T. Fielding, Architectural Styles and the Design of Network-based Software Architectures, 2000. http://www.ics.uci.edu/~fielding/pubs/dissertation/fielding_dissertation.pdf.

[200] M.P. Singh, Interaction-Oriented Programming: Concepts, Theories, and Results on Commitment Protocols, in: A. Sattar, B. Kang (Eds.), AI 2006 Adv. Artif. Intell. 19th Aust. Jt. Conf. Artif. Intell. Hobart, Aust. December 4-8, 2006. Proc., Springer Berlin Heidelberg, Berlin, Heidelberg, 2006: pp. 5–6. doi:10.1007/11941439_3.

[201] T. Coupaye, J.-B. Stefani, Fractal Component-Based Software Engineering, in: M. Südholt, C. Consel (Eds.), Object-Oriented Technol. ECOOP 2006 Work. Read., Springer Berlin Heidelberg, Berlin, Heidelberg, 2007: pp. 117–129. doi:10.1007/978-3-540-71774-4_13.

[202] Chien-Liang Fok, G.-C. Roman, Chenyang Lu, Mobile agent middleware for sensor networks: an application case study, in: IPSN 2005. Fourth Int. Symp. Inf. Process. Sens. Networks, 2005., IEEE, 2005: pp. 382–387. doi:10.1109/IPSN.2005.1440953.

[203] T. Liu, M. Martonosi, Impala, ACM SIGPLAN Not. 38 (2003) 107. doi:10.1145/966049.781516.

[204] H.C.H. Cha, S.C.S. Choi, I.J.I. Jung, H.K.H. Kim, H.S.H. Shin, J.Y.J. Yoo, C.Y.C. Yoon, The RETOS Operating System: Kernel, Tools and Applications, 2007 6th Int. Symp. Inf. Process. Sens. Networks. (2007) 559–560. doi:10.1109/IPSN.2007.4379725.

[205] L. Bauer, A.W. Appel, E.W. Felten, Mechanisms for secure modular programming in Java, Softw. Pract. Exp. 33 (2003) 461–480. doi:10.1002/spe.516.

[206] E.R. Harold, Java network programming, Comput. Math. with Appl. 33 (1997) 147. doi:10.1016/S0898-1221(97)90141-3.

[207] H.-G. Beyer, H.-P. Schwefel, Evolution strategies -- A comprehensive introduction, Nat. Comput. 1 (2002) 3–52. doi:10.1023/A:1015059928466.

[208] R.J. Anthony, Emergence: a paradigm for robust and scalable distributed applications, in: Int. Conf. Auton. Comput. 2004. Proceedings., IEEE, 2004: pp. 132–139. doi:10.1109/ICAC.2004.1301356.

[209] T.H. Labella, M. Dorigo, J.-L. Deneubourg, Division of labor in a group of robots inspired by ants' foraging behavior, ACM Trans. Auton. Adapt. Syst. 1 (2006) 4–25. doi:10.1145/1152934.1152936.

[210] C. Adami, C. Ofria, T.C. Collier, Evolution of biological complexity, Proc. Natl. Acad. Sci. 97 (2000) 4463–4468. doi:10.1073/pnas.97.9.4463.

[211] M. Burgin, E. Eberbach, Recursively Generated Evolutionary Turing Machines and Evolutionary Automata, in: X.-S. Yang (Ed.), Artif. Intell. Evol. Comput. Metaheuristics Footsteps Alan Turing, Springer Berlin Heidelberg, 2013: pp. 201–230. doi:10.1007/978-3-642-29694-9_9.

[212] C. Teuscher, Turing's Connectionism, Springer London, London, 2002. doi:10.1007/978-1-4471-0161-1.

[213] Christof Teuscher, Alan Turing: Life and Legacy of a Great Thinker, Springer Berlin Heidelberg, Berlin, Heidelberg, 2004. doi:10.1007/978-3-662-05642-4.

[214] M. Burgin, E. Eberbach, Evolutionary Turing in the Context of Evolutionary Machines, arXiv. (2013). http://arxiv.org/abs/1304.3762.

[215] E. Eberbach, M. Burgin, Evolution of evolution: Self-constructing Evolutionary Turing Machine case study, in: 2007 IEEE Congr. Evol. Comput., IEEE, 2007: pp. 4599–4605. doi:10.1109/CEC.2007.4425074.

[216] Z. Michalewicz, D.B. Fogel, How to Solve It: Modern Heuristics, Springer Berlin Heidelberg, Berlin, Heidelberg, 2004. doi:10.1007/978-3-662-07807-5.

[217] H.M. Cartwright, An Introduction to Evolutionary Computation and Evolutionary Algorithms, in: Intell. Control Syst. Using Soft Comput. Methodol., 2004: pp. 1–32. doi:10.1007/b13931.

[218] T. Ito, K. Ono, M. Ichikawa, Y. Okuyama, K. Kuroda, Reconfigurable Instruction-Level Parallel Processor Architecture, in: 2003: pp. 208–220. doi:10.1007/978-3-540-39864-6_17.

[219] M. Thornburg, S. Casselman, Transformable computers, in: Proc. 8th Int. Parallel Process. Symp., IEEE Comput. Soc. Press, 2002: pp. 674–679. doi:10.1109/IPPS.1994.288231.

[220] H.A. Chow, H. Alnuweiri, S. Casselman, FPGA-based transformable computers for fast digital signal processing, in: Proc. IEEE Symp. FPGAs Cust. Comput. Mach., IEEE Comput. Soc. Press, 1995: pp. 197–203. doi:10.1109/FPGA.1995.477426.

[221] S. Casselman, M. Thornburg, J. Schewel, Hardware object programming on the EVC1: a reconfigurable computer, in: J. Schewel (Ed.), Proc. SPIE, 1995: pp. 168–176. doi:10.1117/12.221336.

[222] M. Burgin, E. Eberbach, On Foundations of Evolutionary Computation, in: Handb. Res. Artif. Immune Syst. Nat. Comput., IGI Global, 2009: pp. 342–360. doi:10.4018/978-1-60566-310-4.ch016.

[223] E. Eberbach, Toward a theory of evolutionary computation, Biosystems. 82 (2005) 1–19. doi:10.1016/j.biosystems.2005.05.006.

[224] M. Burgin, E. Eberbach, Cooperative combinatorial optimization: Evolutionary computation case study, Biosystems. 91 (2008) 34–50. doi:10.1016/j.biosystems.2007.06.003.

[225] M. BURGIN, Three aspects of super-recursive algorithms and hypercomputation or finding black swans, Theor. Comput. Sci. 317 (2004) 1–11. doi:10.1016/S0304-3975(03)00628-5.

[226] M. Burgin, Super-recursive Algorithms and Modes of Computation, in: ECSAW 2015 - Eur. Conf. Softw. Archit. Work., 2015. doi:10.1145/2797433.2797443.

[227] J.-F. Raskin, An introduction to hybrid automata, in: Handb. Networked Embed. Control Syst., 2005: pp. 491–517. http://www.cmi.ac.in/~madhavan/courses/qath-2015/reading/Raskin_Intro_Hybrid_Automata.pdf.

[228] F. Cassez, J.J. Jessen, K.G. Larsen, J.-F. Raskin, P.-A. Reynier, Automatic Synthesis of Robust and Optimal Controllers – An Industrial Case Study, in: Lect. Notes Comput. Sci. (Including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics), 2009: pp. 90–104. doi:10.1007/978-3-642-00602-9_7.

[229] Hong Fu, Guangyu Tian, Quanshi Chen, Yiding Jin, Hybrid automata of an integrated motor-transmission powertrain for automatic gear shift, in: Proc. 2011 Am. Control Conf., IEEE, 2011: pp. 4604–4609. doi:10.1109/ACC.2011.5991137.

[230] Zhihao Jiang, M. Pajic, R. Mangharam, Cyber-Physical Modeling of Implantable Cardiac Medical Devices, Proc. IEEE. 100 (2012) 122–137. doi:10.1109/JPROC.2011.2161241.

[231] W. Pasillas-Lépine, Hybrid modeling and limit cycle analysis for a class of five-phase anti-lock brake algorithms, Veh. Syst. Dyn. 44 (2006) 173–188. doi:10.1080/00423110500385873.

[232] R. Grosu, C. Klein, B. Rumpe, M. Broy, State Transition Diagrams, in: Concept. Model. Inf. Syst., Springer Berlin Heidelberg, Berlin, Heidelberg, 1996: pp. 299–323. doi:10.1007/978-3-540-39390-0_13.

[233] C. Calcaterra, A. Boldt, Lipschitz Flow-box Theorem, J. Math. Anal. Appl. 338 (2008) 1108–1115. doi:10.1016/j.jmaa.2007.06.001.

[234] E.A. Lee, Cyber Physical Systems: Design Challenges, in: 2008 11th IEEE Int. Symp. Object Component-Oriented Real-Time Distrib. Comput., IEEE, 2008: pp. 363–369. doi:10.1109/ISORC.2008.25.

[235] Gene F. Franklin; J.David Powell; Abbas Emami-Naeini, Feedback Control of Dynamic Systems, 4th ed., Prentice Hall PTR, 2015.

[236] J. Allan, R. Papka, V. Lavrenko, On-line new event detection and tracking, in: Proc. 21st Annu. Int. ACM SIGIR Conf. Res. Dev. Inf. Retr. - SIGIR '98, ACM Press, New York, New York, USA, 1998: pp. 37–45. doi:10.1145/290941.290954.

[237] M. Dao, S. Pongpaichet, L. Jalali, K. Kim, R. Jain, K. Zettsu, A Real-time Complex Event Discovery Platform for Cyber-Physical-Social Systems, in: Proc. Int. Conf. Multimed. Retr. - ICMR '14, ACM Press, New York, New York, USA, 2014: pp. 201–208. doi:10.1145/2578726.2578755.

[238] R. Reiter, Knowledge in action: Logical foundations for specifying and implementing dynamical systems, MIT Press, 2001.

[239] F. Pirri, R. Reiter, Planning with Natural Actions in the Situation Calculus, in: Logic-Based Artif. Intell., Springer US, Boston, MA, 2000: pp. 213–231. doi:10.1007/978-1-4615-1567-8_10.

[240] E.T. Mueller, Event Calculus, in: F. van Harmelen, V. Lifschitz, B. Porter (Eds.), Handb. Knowl. Represent., 2008: pp. 671–708. doi:10.1016/S1574-6526(07)03017-9.

[241] Y.-C. Ho, Introduction to special issue on dynamics of discrete event systems, Proc. IEEE. 77 (1989) 3–6. doi:10.1109/5.21065.

[242] M. Thielscher, Introduction To The Fluent Calculus, Electron. Trans. AI. 2 (1998) 197–192. http://cgi.cse.unsw.edu.au/~mit/Papers/ETAI98.pdf.

[243] J. McCarthy, P.J. Hayes, Some philosophical problems from the standpoint of artificial intelligence, English. 4 (1969) 463–502. doi:10.1.1.85.5082.

[244] F. Pirri, R. Reiter, Some contributions to the metatheory of the situation calculus, J. ACM. 46 (1999) 325–361. doi:10.1145/316542.316545.

[245] H.J. Levesque, R. Reiter, Y. Lespérance, F. Lin, R.B. Scherl, GOLOG: A logic programming language for dynamic domains, J. Log. Program. 31 (1997) 59–83. doi:10.1016/S0743-1066(96)00121-5.

[246] J. McCarthy, Actions and other events in situation calculus, Kr. (2002). http://www-formal.stanford.edu/jmc/sitcalc.pdf.

[247] A. Finzi, F. Pirri, R. Reiter, Open world planning in the situation calculus, in: Proc. 7th Conf. Artif. Intell. 12th Conf. Innov. Appl. Artif. Intell. (IAAI-00, 1995: pp. 754–760. http://dl.acm.org/citation.cfm?id=647288.721250.

[248] Lichuan Liu, S.M. Kuo, M. Zhou, Virtual sensing techniques and their applications, in: 2009 Int. Conf. Networking, Sens. Control, IEEE, 2009: pp. 31–36. doi:10.1109/ICNSC.2009.4919241.

[249] B. Lin, B. Recke, J.K.H. Knudsen, S.B. Jørgensen, A systematic approach for soft sensor development, Comput. Chem. Eng. 31 (2007) 419–425. doi:10.1016/j.compchemeng.2006.05.030.

[250] W.T.L. Teacy, J. Nie, S. McClean, G. Parr, S. Hailes, S. Julier, N. Trigoni, S. Cameron, Collaborative Sensing by Unmanned Aerial Vehicles, Communication. (2009) 13–16. http://www.cs.ox.ac.uk/files/3063/atsn.pdf.

[251] A. Krohn, T. Zimmer, M. Beigl, C. Decker, Collaborative Sensing in a Retail Store Using Synchronous Distributed Jam Signalling, in: Int. Conf. Pervasive Comput. 2005, 2005: pp. 237–254. doi:10.1007/11428572_15.

[252] NECSI, Evolutionary System Model, (n.d.). http://www.necsi.edu/projects/mclemens/evomodel.gif (accessed August 12, 2016).

[253] C. Brooks, C.-H. Cheng, T.H. Feng, E.A. Lee, Reinhard von Hanxleden, Model Engineering using Multimodeling, in: 1st Int. Work. Model Co-Evolution Consitency Manag., 2008: pp. 1–38. https://chess.eecs.berkeley.edu/pubs/486/Multimodeling.pdf.

[254] K. Balasubramanian, A. Gokhale, G. Karsai, J. Sztipanovits, S. Neema, Developing Applications Using Model-Driven Design Environments, Computer (Long. Beach. Calif). 39 (2006) 33–40. doi:10.1109/MC.2006.54.

[255] G. Karsai, J. Sztipanovits, A. Ledeczi, T. Bapty, Model-integrated development of embedded software, Proc. IEEE. 91 (2003) 145–164. doi:10.1109/JPROC.2002.805824.

[256] J. Eker, J.W. Janneck, E.A. Lee, Jie Liu, Xiaojun Liu, J. Ludvig, S. Neuendorffer, S. Sachs, Yuhong Xiong, Taming heterogeneity - the Ptolemy approach, Proc. IEEE. 91 (2003) 127–144. doi:10.1109/JPROC.2002.805829.

[257] Y. Tan, S. Goddard, L.C. Pérez, A prototype architecture for cyber-physical systems, ACM SIGBED Rev. 5 (2008) 1–2. doi:10.1145/1366283.1366309.

[258] T. Noergaard, Embedded Processors, in: Embed. Syst. Archit., Elsevier, 2005: pp. 129–221. doi:10.1016/B978-075067792-9/50008-X.

[259] European Parliament, Next Generation Networks, (2009).

https://arxiv.org/ftp/arxiv/papers/1012/1012.2524.pdf.

[260] E. Wu, Y. Diao, S. Rizvi, High-performance complex event processing over streams, in: Proc. 2006 ACM SIGMOD Int. Conf. Manag. Data - SIGMOD '06, ACM Press, New York, New York, USA, 2006: p. 407. doi:10.1145/1142473.1142520.

[261] P. Johannesson, E. Perjons, Design Principles for Application Integration, in: 12th Conf. Adv. Inf. Syst. Eng., 2000: pp. 212–231. doi:10.1007/3-540-45140-4_15.

[262] N. Bencomo, A. Belaggoun, Supporting Decision-Making for Self-Adaptive Systems: From Goal Models to Dynamic Decision Networks, in: Lect. Notes Comput. Sci. (Including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics), 2013: pp. 221–236. doi:10.1007/978-3-642-37422-7_16.

[263] S. Russel, P. Norvig, Artificial Intelligence: A Modern Approach, 2nd ed., Pearson Education, 2003.

[264] M. Missikoff, Y. Charalabidis, R. Jardim-Gonçalves, K. Popplewell, Future Internet Enterprise Systems (FInES): Research Roadmap, 2013. http://cordis.europa.eu/fp7/ict/enet/documents/fines-researchroadmap-final-report.pdf.

[265] Z. Alliance, Network Device : Gateway Specification Revision 35, ReVision. (2011) 301. http://contech.suv.ac.kr/contech/courses/11h2wsn/115552r00ZB_BoD-ZigBee_Network_Device_-_ZigBee_Gateway_standard_version_1.pdf.

[266] F. Luis-Ferreira, S. Ghimire, R. Jardim-Goncalves, Brain inspired health monitoring supported by the cloud, in: IFIP Adv. Inf. Commun. Technol., 2015: pp. 273–281. doi:10.1007/978-3-319-16766-4_30.

[267] S. Ghimire, R. Melo, J. Ferreira, C. Agostinho, R. Goncalves, Continuous Data Collection Framework for Manufacturing Industries, in: Lect. Notes Comput. Sci. (Including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics), 2015: pp. 29–40. doi:10.1007/978-3-319-26138-6_5.

[268] K.S. Oie, T. Kiemel, J.J. Jeka, Human multisensory fusion of vision and touch: detecting non-linearity with small changes in the sensory environment, Neurosci. Lett. 315 (2001) 113–116. doi:10.1016/S0304-3940(01)02348-5.

[269] Roland Berger Strategy Consultants, Industry 4.0 The new industrial revolution How Europe will succeed, 2014. https://www.rolandberger.com/publications/publication_pdf/roland_berger_tab_industry_4_0_20140403.pdf.

[270] P. Petrali, Smart Factory Experimentation Report WP4 Smart Factory Trials : instantiation , adaption , experimentation, 2015. http://cordis.europa.eu/docs/projects/cnect/4/604674/080/deliverables/001-D43SmartFactoryExperimentationReport.pdf.

[271] R. Sanguini, J. Sola, Digital Factory Experimentation Report WP5 Digital Factory Trials : instantiation , adaption , experimentation, 2015. http://cordis.europa.eu/docs/projects/cnect/4/604674/080/deliverables/001-D53FITMANDigitalFactoryExperimentationReportv10.pdf.

[272] FITMAN-Consortium, FITMAN Smart-Digital-Virtual Factory Trials Experiences, 2015. http://tinyurl.com/zebufhz.

[273] T. Bulbul, C.J. Anumba, J. Messner, A System of Systems Approach to Intelligent Construction Systems, in: Comput. Civ. Eng., American Society of Civil Engineers, Reston, VA, 2009: pp. 22–32. doi:10.1061/41052(346)3.

[274] W. Shen, Q. Hao, H. Mak, J. Neelamkavil, H. Xie, J. Dickinson, R. Thomas, A. Pardasani, H. Xue, Systems integration and collaboration in architecture, engineering, construction,

and facilities management: A review, Adv. Eng. Informatics. 24 (2010) 196–207. doi:10.1016/j.aei.2009.09.001.

[275] M.R. Endsley, Theoretical Underpinnings of Situation Awareness: A Critical Review, in: M.R. Endsley, D.J. Garland (Eds.), Situat. Aware. Anal. Meas., Lawrence Erlbaum Associates, 2000: pp. 3–32.

[276] J. Llinas, C. Bowman, G. Rogova, A. Steinberg, Revisiting the JDL data fusion model II, Sp. Nav. Warf. Syst. Command. 1 (2004) 1–14. http://oai.dtic.mil/oai/oai?verb=getRecord&metadataPrefix=html&identifier=ADA525721 (accessed July 25, 2014).

[277] V. Gorodetsky, O. Karsaev, V. Samoilov, On-line update of situation assessment: A generic approach, Int. J. Knowledge-Based Intell. Eng. Syst. 9 (2005) 351–365. doi:10.3233/KES-2005-9410.

[278] N. Baumgartner, W. Retschitzegger, Towards a Situation Awareness Framework Based on Primitive Relations, in: 2007 Information, Decis. Control, IEEE, 2007: pp. 291–295. doi:10.1109/IDC.2007.374565.

[279] A.-C. Boury-Brisset, Ontology-based approach for information fusion, in: Sixth Int. Conf. Inf. Fusion, 2003. Proc., IEEE, 2003: pp. 522–529. doi:10.1109/ICIF.2003.177491.

[280] M.R. Endsley, Toward a Theory of Situation Awareness in Dynamic Systems, Hum. Factors J. Hum. Factors Ergon. Soc. 37 (1995) 32–64. doi:10.1518/001872095779049543.

[281] C.J. Matheus, K.P. Baclawski, M.M. Kokar, Derivation of ontological relations using formal methods in a situation awareness scenario, in: B. V. Dasarathy (Ed.), Proc. SPIE, 2003: p. 298. doi:10.1117/12.488441.

[282] A. Majumdar, S. Hadrien, The ActionPlanT Vision for Manufacturing 2 . 0, 2011. http://cordis.europa.eu/fp7/ict/micro-nanosystems/docs/fof-evaluators/actionplan-vision_en.pdf.

[283] E. Commission, Factories of the Future: Multi-Annual Roadmap for the Contractual PPP Under HORIZON 2020, 2013. http://www.effra.eu/attachments/article/129/Factories of the Future 2020 Roadmap.pdf.

[284] FITMAN-Consortium, FITMAN Reference Architecture, (2013) 1–60. http://cordis.europa.eu/fp7/ict/netinnovation/deliverables/fitman/fitman-d15.pdf.

[285] M.M. Kokar, M.R. Endsley, Situation Awareness and Cognitive Modeling, IEEE Intell. Syst. 27 (2012) 91–96. doi:10.1109/MIS.2012.61.

[286] M. Martinez, S. Sequoiah-Grayson, Logic and Information, Stanford Encycl. Philos. (2014).

[287] OMG, Semantics of Business Vocabulary and Business Rules (SBVR), OMG, 2013. http://www.omg.org/spec/SBVR/1.2/PDF.

[288] M. Hermann, T. Pentek, B. Otto, Design Principles for Industrie 4.0 Scenarios, in: 2016 49th Hawaii Int. Conf. Syst. Sci., IEEE, 2016: pp. 3928–3937. doi:10.1109/HICSS.2016.488.

[289] F.L.L. Ferreira, Knowledge management framework based on brain models and human physiology, 2015. http://run.unl.pt//handle/10362/15445.

[290] T. Nodehi, A new MDA-SOA based Framework for Intercloud Interoperability, 2014. https://run.unl.pt/bitstream/10362/14377/1/Nodehi_2014.pdf.

[291] T. Nodehi, S. Ghimire, R. Jardim-Goncalves, A. Grilo, On MDA-SOA based Intercloud Interoperability framework, J. Comput. Methods Soc. Sci. 1 (2013). http://www.fitman-fi.eu/publications/articles/on-mda-soa-based-intercloud-interoperability-framework.

[292] S. Ghimire, R. Jardim-Goncalves, A. Grilo, M. Beca, Framework for inter-operative e-Procurement marketplace, in: Proc. 2013 IEEE 17th Int. Conf. Comput. Support. Coop. Work Des., IEEE, 2013: pp. 459–464. doi:10.1109/CSCWD.2013.6581006.

[293] S. Ghimire, R. Jardim-Goncalves, A. Grilo, Framework for catalogues matching in procurement e-marketplaces, Inf. Syst. Technol. (CISTI), 8th Iber. Conf. (2013) 1–6. http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6615757.

[294] S. Ghimire, F. Luis-Ferreira, T. Nodehi, R. Jardim-Goncalves, IoT based situational awareness framework for real-time project management, Int. J. Comput. Integr. Manuf. 0 (2016) 1–10. doi:10.1080/0951192X.2015.1130242.

[295] F. Luis-Ferreira, S. Ghimire, R. Jardim-Goncalves, Towards Human-Centric Healthcare supported by Emotional Assessment, in: ICE Conf. 2015, 2015.

11. APPENDIX A (ALGORITHMS)

In this chapter we will provide pseudo code of some of the algorithms for implementing different functionalities defined in the dissertation.

11.1. ALGORITHM FOR CONCEPTS SIMILARITY

Table 11-1 Pseudo code for Semantic Distance Solving Algorithm

Input: two concepts C1, C2
Output: semantic distance: Sem_Dis(C1,C2)
1. For two ontological concepts: C1, C2 2. If C1, C2 are the same concept 3. Sem_Dis(C1, C2)=0 4. Else if there exists the direct path relation between C1 and C2 5. Sem_Dis(C1, C2) = w[sub(C1, C2)] 6. Else if there exists the indirect path relations between C1 and C2 7. $Sem\_Dis(C1, C2) = \sum_{C \in SPath(C1,C2)} w_c[sub(C1, C2)]$ Where, SPath denotes the shortest path between C1 and C2 8. Else 9. Sem_Dis(C1, C2) = min{Sem_Dis(C1,C0)} + min{Sem_Dis(C2,C0)} where C0 is the root.

11.2. GA FOR PLAN OPTIMIZATION

Table 11-2 Pseudo code for Genetic Algorithm for plan optimization

Initialization: 1. Set t=0 2. Generate initial population P_t using Plan Model 3. Load fitness function C_{opt}
4. Evaluate the population P_t 5. While the stopping criteria is not met DO 6. { 7. Select some elements from P_t to copy to P_{t+1} 8. Perform crossover on selected P_t and move them to P_{t+1} , according to the defined value for crossover probability 9. Perform mutation on selected P_t and move them to P_{t+1} , according to the defined value for mutation probability 10. Evaluate the new population P_{t+1} 11. $P_t = P_{t+1}$ 12. }

11.3. ALGORITHM FOR PARETO FRONT OPTIMIZATION

Table 11-3 Pseudo code for Pareto front optimization algorithm

1. Initialize open and closed lists ; list of plans to be optimized are in open lists, while

optimized plans are in closed lists
2. Put the starting node in the open list ; starting node is the first activity of the plan
3. Define f, the cost function ; the optimization function C_{opt}
4. While the open list is not empty
5. $q \leftarrow n$ where n is the node on open list with smallest f (evaluation function)
6. Remove q (or n) from open list
7. Generate $q's$ 8 successors, set their parents to q ; 8 is a randomly select constant higher the value of this constant higher is the complexity
8. For each successor
9. If successor is a goal, then stop
10. Else
11. successor.g ← q.g + distance between successor and q where g is heuristic the cost to reach the node n.
12. successor.h ← distance from successor to goal where h is the heuristic function that gives the estimated cost of the cheapest path from a node to the goal state
13. scoreMatrix(successor) ← [successor.g, successor.h]
14. End For
15. q ← Calculate Pareto front of scoreMatrix
16. If multiple points on Pareto front
17. Normalize scoreMatrix
18. Push q to the closed list
19. End While

Calculate Pareto Font is performed by the submodule

1. Let i=1.
2. Add A_i to the Pareto frontier where A_i are the alternatives in increasing order of cost
3. Find smallest j> i such that cost(A_j)>cost(A_i)
4. If no such j exists, stop. Otherwise let i=j and repeat from step 2.